VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Využití kvantové informace v analýze textu

Usage of Quantum Information in Text Processing

2018 Bc. Martin Kössler

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Kössler**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Využití kvantové informace v analýze textu
Usage of Quantum Information in Text Processing

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je implementovat nástroj pro experimentování se zpracováním textu za pomocí kvantové logiky a kvantové informace. Nástroj bude schopen provádět hledání v dokumentech pomocí definovaných pravidel založených na kvantové informaci.

Práce bude obsahovat:
1. Seznámení s problematikou.
2. State of the art.
3. Podrobný popis zvoleného/zvolených algoritmů.
4. Experimenty, měření, vyhodnocení (možno použít tabulky a grafy).
5. Závěr - zhodnocení výsledků.

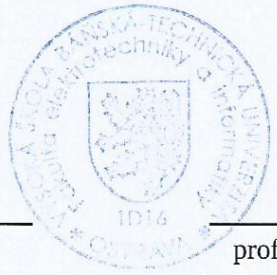
Seznam doporučené odborné literatury:

[1] M. Melucci, R. Baeza-Yates, Advanced Topics in Information Retrieval, Springer, 2011.
[2] M. Hayashi, Quantum Information: An Introduction, Springer, 2006.
[3] D. Widdows and S. Peters. Word Vectors and Quantum Logic: Experiments with negation and disjunction. Eighth Mathematics of Language Conference, Bloomington, Indiana, June 20-22, 2003, pages 141-154.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jan Platoš, Ph.D.**

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018 .....

Rád bych poděkoval vedoucímu mé diplomové práce doc. Ing. Janu Platošovi, Ph.D. za jeho odbornou pomoc a trpělivost při vytváření této diplomové práce. Dále bych chtěl poděkovat všem, kteří mě při vytváření této práce podporovali.

Abstrakt

Diplomová práce se zabývá využitím kvantové informace v analýze textu. Práce je rozdělena na teoretickou a praktickou část. Teoretická část se zabývá popisem kvantové logiky a její interpretací v textu, dále pak procesem získávání informací a zpracováním přirozeného jazyka. Jsou zde definovány modely a operace pro vyhledávání dotazů. Praktická část obsahuje popis experimentů, které mají za účel vyzkoušet vyhledávání dotazů.

Klíčová slova: Analýza textu, Kvantová logika, Podobnost slov, Content Bearing Words

Abstract

The diploma thesis deals with Usage of quantum information in text processing. The thesis is divided into theoretical and practical part. Theoretical part explains quantum logic and its interpretation in text, as well as information retrieval and natural-language proccessing. The theoretical part also presesnts definitions of models and operation of query searching. The practical part then describes the experiments which were performed to prove the statement of query searching previously mentioned.

Key Words: Text processing, Quantum logic, Words similarity, Content Bearing Words

Obsah

Seznam použitých zkratek a symbolů

CBW	–	Content Bearing Words
CW	–	Context window - kontextové okno
PS	–	Porter Stemmer
LSA	–	Latent Semantic Analysis
Sim	–	Similarity - podobnost
SL	–	Stoplist
SM	–	Similarity Matrix - matice podobnosti
SVD	–	Singular Value Decomposition
Term	–	Slovo, Termín
T-D	–	Term-Document
TF-IDF	–	Term Frequency - Inverse Document Frequency

Seznam obrázků

Seznam tabulek

1 Úvod

Lidská řeč je jedna z nejsložitějších forem komunikace, plná nejednoznačností a výjimek. Při komunikaci se setkáváme s celou řadou nedorozumění, které jsou často způsobeny slovy o více než jednom významu. O to náročnější je proces zpracování přirozeného jazyka pro počítačové systémy, které musí obsahovat různé slovníky těchto, pro něj matoucích, slov.

Tato diplomová práce se bude zabývat hledáním významů slov za použití kvantové logiky, která na rozdíl od klasické logiky není deterministická. Řídí se pouze na základě pravděpodobnosti a dopředu nevíme jaký bude výsledek. Tímto se velmi liší od klasické logiky, kde předpokládáme, že pokud víme všechny potřebné informace stávajícího stavu, můžeme předpovědět i stav budoucí.

Bude použito několik typů metod, které jsou zaměřeny na úpravu hledaného dotazu, i na typy porovnávání.

Teoretická část této práce je rozdělena na dvě části, z nichž první je zaměřena na rozdíly mezi kvantovou a Booleovskou logikou. Tato část se zabývá pojmy nutnými k pochopení provedených experimentů a jejich souvislost s kvantovou logikou. Důležité je rovněž pochopit podobnosti kvantové a Booleovské (neboli klasické) logiky, jelikož i na nich staví velká část této práce. Bude zde také popsán princip superpozice a jak by se mohl dát využít ve zpracování textu.

Druhá teoretická část se zabývá procesem získávání informací. Je zde popsán základní postup zpracování textů přirozeného jazyka. V dalších podkapitolách budou popisovány statistické a matematicko-lingvistické metody, jak tato relevantní slova v textu identifikovat a extrahovat je do seznamu unikátních slov.

Teoretická část popisuje maticové modely, které slouží pro převod slov na vektory. Mezi zajímavé matice patří *Content Bearing Words*, jejíž hlavní účel je nalezení významu slov a matice podobnosti, kterou jsem navrhl tak, aby reprezentovala kvantový stav. Mimo jiné zde bude popsáná metrika podobnosti a také způsob normalizace, které platí pro všechny vektory (a matice) v této práci.

Poslední teoretická část obsahuje popis tvorby dotazů. Z nástrojů klasické logiky vytváříme operace logiky kvantové, které slouží pro modifikaci těchto dotazů. Jmenovitě je to operace NOT , která umožňuje změnit význam slova pouze na základě vektorových operací. Další operace OR umožňuje například určit míru podobnosti vektoru s maticí.

Praktická část je rozdělena na tři kapitoly. První kapitola se zabývá popisem experimentů pro získávání informací z dokumentů, především rovnicí kvantového stavu (v každé matici řádky jako vektory slov) a její interpretací. Zároveň je zde popsaná automatická extrakce termínů do slovníku. Jelikož byly pro experimenty zvoleny dokumenty v angličtině, jsou zde popsány metody a výjimky, které slouží speciálně pro tento jazyk. V několika experimentech bude předvedena účinnost filtrů v lexikální analýze a jaký má vliv na výsledky hledání dotazů. Následně bude popsán postup tvorby dotazu. Pro vyhledávání bylo použito několik přístupů, hledání podob-

nosti dotazu se slovy se zaměřuje na porovnávání jednotlivých vektoru slov s dotazem. Druhou možností je podobnost dotazu s celým dokumentem.

Druhá kapitola praktické části se zabývá systémem pro zpracování přirozeného jazyka a získávání informací, ve kterém se lze dotazovat různými způsoby a hledat podobnosti mezi slovy, skupinou slov a dokumenty.

Poslední kapitola praktické části popisuje výsledky všech experimentů, které byly provedeny. Nejdříve jsou zde popsána testovací data a dále vliv filtrování při extrakci slov do slovníku, kde jsou naznačeny výhody i nevýhody použití filtrovacích metod. Další experiment se zabývá vlivem velikosti kontextového okna v matici *Content Bearing Words*. Výsledky hledání jsou vyhodnoceny pomocí již zmíněných metod. Jsou zde uvedeny názorné příklady jak metody fungují a jak výsledky správně interpretovat.

2 Klasická a kvantová logika

2.1 Booleova logika

Booleova neboli klasická logika je ideální pro počítačové technologie. Vychází z jednoduché myšlenky, že výraz jako celek je buďto pravdivý nebo nepravdivý, což se velice dobře hodí v binární reprezentaci, kde máme na výběr pouze 0 a 1. Výrazy nemusí být jednoduchého charakteru jako například *'Právě prší.'*, kde jednoduše rozhodneme, zda je výraz pravdivý či nikoli, ale mohou být i složitého charakteru, například *'Právě prší, nesvítí slunce a je zataženo.'*. Tyto výrazy lze vždy rozložit na triviální, což je činí užitečnou metodu zpracování informací pro počítačové technologie. Tato logika má definované operace:

- hodnoty TRUE / FALSE
- negace NOT
- konjunkce AND
- disjunkce OR

Platí zde komutativní a distributivní zákony.

2.1.1 Komutativita

Komutativita je vlastnost binární operace, která spočívá v tom, že u ní nezávisí na pořadí jednotlivých operandů. Například binární operace $*$ je na množině S komutativní, jestliže platí:

$$\forall x, y \in S : x * y = y * x$$

2.1.2 Distributivita

Distributivita je vlastnost binární operace, jež vůči jiné binární operaci říká, že můžeme tuto operaci distribuovat přes jinou operaci. Například binární operace $*$ je na množině S distributivní vůči operaci $+$, jestliže platí:

$$\forall x, y \in S : x * (y + z) = (x * y) + (x * z) \wedge (y + z) * x = (y * x) + (z * x)$$

2.2 Kvantová logika

Kvantová logika se více přibližuje lidskému přemýšlení. Výraz nemá jeden pravdivý stav, ale všechny najednou, pouze se odlišují různými pravděpodobnostmi, že danému výrazu odpovídají. To vyplývá z principu superpozice, z něhož také vyplývá, že u kvantové logiky neplatí distributivní zákon.

2.2.1 Princip superpozice

Tento princip vychází z kvantové mechaniky. Ta udává, že pravidla, která platí v mikrosvětě (velikost atomů a menší) neboli v kvantovém světě, se liší od těch, která můžeme pozorovat v makrosvětě. Jedno ze základních pravidel makrosvěta je, že každý objekt má pouze jednu polohu a rychlost, je tedy nemožné cestovat z Německa do Velké Británie autem rychlostí 100km/h a současně letět z Evropy do Austrálie osminásobnou rychlostí. Překvapivě se ukázalo, že toto fundamentální pravidlo pro kvantový svět neplatí, a objekty tedy mohou být na více místech a dělat více věcí současně.[1]

Celá klasická fyzika je postavena na myšlence tzv. determinismu. Základním principem determinismu je to, že budoucnost je předvídatelná a že jediné, co je třeba k předvídání budoucího vývoje vesmíru je dostatek informací o přítomnosti. Můžeme například předvídat, kdy bude další zatmění Slunce, když máme dostatek informací o pohybu Měsíce. Další myšlenkou determinismu je to, že stejné podmínky vedou ke stejným výsledkům. Když například vystřelíme z pistole dvě identické kulky za shodných podmínek (stejná teplota, stejný směr atd.), obě dopadnou na stejné místo. Kvantový svět však funguje naprosto odlišně. Kdybychom místo kulek stříleli například elektrony, mohlo by se stát, že by každý z nich dopadl jinam a pod jinou rychlostí, i když by podmínky při výstřelu byly identické.[1]

Chování elektronů (viz výše) je důsledkem stěžejního jevu kvantové fyziky - principu superpozice stavů. Princip superpozice říká, že není-li objekt pozorován, existuje ve všech možných stavech současně - je v superpozici. Tato superpozice je kombinace všech stavů, ve kterých by daný objekt teoreticky mohl být. To znamená, že dokud částice není pozorována, může mít více rychlostí najednou, nebo být na více místech současně.[1]

Toto chování může znít velice podivně, dokud však není brána v potaz tzv. vlnová funkce. Vlnovou funkci (Ψ) obsahuje každý kvantový objekt. Časový a prostorový vývoj této funkce popisuje složitá rovnice, kterou v roce 1925 definoval rakouský fyzik Erwin Schrödinger. Ve funkci jsou uloženy všechny vlastnosti (hybnost, poloha atd.) daného kvantového objektu a tento soubor všech vlastností je nazýván kvantový stav a značí se takto: $|\psi\rangle$. Pokud tedy vlnová funkce existuje, není poloha tohoto objektu plně definována a takový objekt se v podstatě nachází všude, kde se nachází jeho vlnová funkce. Aby měl kvantový objekt jasně určenou pozici, musíme vlnovou funkci zredukovat na jediný stav (jedna rychlost, jedna poloha). Toho docílíme prostým pozorováním objektu (tj. měřením jeho vlastností), kdy dojde k tzv. kolapsu superpozice. To zajistí, že není možné pozorovat objekt na více místech najednou, jelikož samotným aktem pozorování je superpozice zničena. Tudíž pozorováním nejenže jeho vlastnosti zjišťujeme, ale i definujeme.[1]

Vyvstává však otázka, jakým způsobem si objekt vybírá vlastní stav, ve kterém se v případě pozorování bude nacházet. Tento proces je založen na pravděpodobnosti. Pravděpodobnost, s jakou kvantový objekt skončí v určitém vlastním stavu, je určena jeho vlnovou funkcí. Pravděpodobnost, že daný kvantový systém skončí v určitém vlastním stavu, je určena jako druhá

mocnina absolutní hodnoty amplitudy pravděpodobnosti. Každý kvantový objekt může být v superpozici libovolného množství vlastních stavů, přičemž každému z těchto stavů je přiřazena určitá pravděpodobnostní hodnota (tedy z intervalu <0;1>). Vždy však platí, že součet pravděpodobnostních hodnot všech vlastních stavů kvantového objektu v superpozici je roven jedné. Pravděpodobnost, že bude kvantový objekt nalezen v jednom z vlastních stavů superpozice, je vždy rovna 100%.[1]

$$\sum_{k}^{n} |c_k|^2 = 1$$

c_k ... hodnota amplitudy pravděpodobnosti,

$$|\psi\rangle = \sum_{k}^{n} c_k |\Phi_k\rangle$$

Φ_k ... vlastní stav kvantového objektu,

2.2.2 Přirozený jazyk

Při vyslovení věty *'Přijď v 7 nebo 7:30.'* předpokládáme z hlediska přirozeného jazyka, že je v pořádku, když přijdeme v 7:15, což ale v Booleovské logice není, jelikož jsme nepřišli ani v jeden daný čas. Kvantová logika proto bere v potaz, že se jedná o interval, a proto vyhodnotí náš příchod v 7:15 jako přijatelný. Nicméně, pokud je položen dotaz *'Dáš si jablko nebo banán?'*, z kvantové logiky by byla přijata i odpověď *'pomeranč'*, pokud předpokládáme, že je v množině ovoce někde mezi jablkem a banánem. Nelze proto jasně říct, která logika je pro zpracování přirozeného jazyka lepší.[3]

3 Získávání informací

Získávání informací je vědecká disciplína zaobírající se hledáním informací v dokumentu, vyhledáváním samotných dokumentů a také vyhledáváním metadat, která popisují data a databázi textů, obrázků či zvuků.[4]

3.1 Přehled

Proces získávání informací začíná momentem, kdy uživatel zadá dotaz do systému. Dotazy jsou formální zápisy požadavku na informace, příkladem můžou být textové řetězce ve webových vyhledávačích. V procesu získávání informací nemusí dotaz jednoznačně identifikovat právě jeden objekt v souboru. Namísto toho může dotaz odpovídat vícero objektům více či méně relevantních, a to na základě jejich ohodnocení podle shody s dotazem. Tímto se liší získávání informací od databázového hledání.[4]

3.1.1 Analýza textu

Analýza textu je činnost, při které se zpracovávají dokumenty, jejich texty a získávají se z nich informace. Předtím, než začne fáze získávání informací, musí dokumenty projít procesem přípravy, mezi které patří automatická extrakce slov a jejich filtrace.[4] Lze také zahrnout úpravy slov na základní tvar neboli lematizace.

3.1.2 Homograf

Při zpracování přirozeného jazyka se setkáváme s problémem slov, které se píší stejně, avšak mají jiný význam. Tato slova se nazývají homografy. Angličtina jich obsahuje velké množství (např. *'lead'* - 'olovo' nebo 'vést') a mohou se zde objevit názvy (např. *'Apple'* - 'jablko' nebo IT firma). V češtině se častěji objevují homonyma, mezi něž patří homografy, tedy slova, která se graficky zaznamenávají stejně, ale vyslovují se jinak, příkladem může být slovo *panický* [panicki:] a [paňicki:]. Pokud chceme zjistit, jaký má homograf význam, musíme znát jeho kontext.

3.2 Automatická extrakce

Účelem automatické extrakce je vybrat přímo z textu dokumentu taková slova, která vyjadřují jeho obsah.[2]

3.2.1 Lexikální analýza

Jedná se o proces, při kterém se identifikují jednotlivá slova a sousloví v textu dokumentu. Pro většinu jazyků jsou jednotlivá slova v textu nejčastěji rozpoznávána pomocí mezer, i když u některých typů slov není tato identifikace zcela jednoduchá nebo jednoznačná. Obtížnější je např. určování zkratek, v nichž je tečku ukončující zkratku nutno odlišit od tečky ukončující

větu (např. *'atd.'*). Problém představují také výrazy jež jsou spojeny tzv. spojovníkem (např. *'Frýdek-Místek', 'česko-anglický'*), u kterých je třeba rozhodnout, zda je lze chápat jako jedno slovo nebo dvě samostatná slova. Samostatný problém při identifikaci slov představují také číslice, u kterých je třeba stanovit, zda budou zpracovány jako samostatná slova nebo závislé prvky (např. *'1. místo'*) nebo zda budou z analýzy a dalšího zpracování zcela vypuštěny.[2]

3.2.2 Negativní slovník

Negativní slovník je označován také jako seznam stopslov. Pokud mluvíme o stopslovu, myslíme tím neplnovýznamové nebo nespecifické slovo, které nenese žádný význam (např. spojky, předložky, částice, mluvnické členy apod.). Tato slova je třeba odstranit, aby do výsledku nevnášela šum. Seznam stopslov lze vytvořit několika způsoby:[2]

1. volbou druhů slov, které nenesou žádný význam a slouží pouze pro syntaktické účely (např. spojky, předložky, částice apod.)[2] Tento slovník je v praxi předefinovaný a jeho velikost závisí na námi zvolené přísnosti. Pokud je tato metoda spojena s procesem lematizace (kapitola 3.2.3), nemusí seznam obsahovat všechny tvary daných stopslov.

2. volbou slov s vysokou absolutní nebo relativní frekvencí výskytu v textu dokumentu; vychází se z empiricky ověřeného předpokladu, že neplnovýznamová a nespecifická slova mají podstatně vyšší frekvenci v textu než plnovýznamová slova. Nevýhoda metody spočívá v tom, že mezi frekventovanými slovy se může vyskytovat i důležitý termín, který popisuje dokument.[2]

3. volbou krátkých slov; vychází z předpokladu, že neplnovýznamová slova jsou krátká.[2] Tato slova nemusejí být obsažena v seznamu, jsou pouze odfiltrována.

3.2.3 Lematizace

Protože se slova a sousloví vyskytují v textu v různých tvarech daných jejich číslem, flexí či jinými gramatickými kategoriemi, je žádoucí slova redukovat na jejich základní tvary, resp. kmeny nebo kořeny. Program, který provádí lematizaci, se nazývá lematizátor.[2]
Lematizaci lze provádět pomocí:

1. slovníku kmenů nebo kořenů; výhodou této metody je minimální chybovost, nevýhodou rozsáhlost slovníku a jeho případné omezení na specifický obor.[2]

2. odstranění afixů; tzn. sufixů (přípon) a prefixů (předpon). Jedná se o nejčastěji užívanou metodu s tím, že příslušný algoritmus je obvykle schopen zohledňovat i nepravidelnou flexi (např. hlasová změny - *'soli' x 'sůl'*). Afixy mohou být odstraňovány na základě seznamů sufixů a prefixů nebo na základě pravidel, podle kterých jsou konkrétní afixy generovány.[2]

3. statisticky na základě variety po sobě následujících písmen ve slově (*letter successor variety stemmers)*), kdy se pomocí frekvence jednotlivých shluků písmen stanovuje, zda se jedná

o prefix, kořen nebo sufix. Tato metoda je nezávislá na jazyce a dokáže pružně zohledňovat nové dokumenty v databázi, nedokáže však rozlišit inflexní a derivační (slovotvorné) afixy.[2]

3.2.4 Stematizace

Stematizace je zjednodušený proces lematizace. Pracuje pouze s daným slovem a pokouší se ho zjednodušit na základní tvar, neboli kmen slova. Nástroj pro stematizaci, tzv. stemmer, nemá k dispozici slovník kmenů a kořenů slov, jako tomu bylo v procesu lematizace, většinou pouze pracuje s afixy (předponami a příponami), ať už jejich odstraněním nebo změnou. K tomu, jakou provést změnu afixu, algoritmus většinou používá slovník, kde se definují tyto změny.[4] Příklady algoritmů pro anglický jazyk:

- Porter Stemmer
- Porter2 Stemmer
- Snowball

3.2.5 Algoritmus Porter Stemmer

Porterův stematizovací algoritmus (nebo *'Porter stemmer'*) je proces pro odstraňování běžných morfologických a inflexních zakončení slov v angličtině. Jeho hlavní použití je součástí procesu normalizace slova, který se obvykle provádí při vytváření systémů pro získávání informací.[6]

Algoritmus byl přepsán do mnoha programovacích jazyků (C, Java, C#) a autor ho dal k dispozici pro veřejnost.[6]

3.3 Modely

Pro efektivní získávání relevantních dokumentů pomocí strategií získávání informací jsou dokumenty většinou transformovány do vhodné reprezentace. Každá strategie získávání informací obsahuje konkrétní model pro reprezentaci dokumentů.[5]

- Množinové modely reprezentující dokumenty jako množiny slov či frází. Podobnosti jsou obvykle odvozeny z množinových operací těchto struktur.[5]
  - Standardní Booleovský model
  - Rozšířený Booleovský model
  - Fuzzy retrieval
- Algebraické modely reprezentují dokumenty a dotazy jako vektory, matice nebo n-tice. Skalární hodnoty potom odpovídají podobnosti vektoru dotazu a vektoru dokumentu.[5]
  - Vektorový model

  - Zobecněný vektorový model
  - Latentní sémantická analýza (LSA)

- Pravděpodobnostní modely přistupují k procesu výběru dokumentu jako v pravděpodobnostním odvozování. Podobnost je vypočítána z pravděpodobnosti, že dokument odpovídá danému dotazu. V těchto modelech se často využívají tvrzení z teorie pravděpodobnosti.[5]
  - Binárně nezávislý model
  - Odvozování nejistoty
  - Lingvistická analýza
  - Latentní Dirichletova alokace

3.3.1 Term-Document

Term-Document (T-D) je vektorový model, který využívá jedinou matici pro ukládání informací o dokumentech a slovech. Tato matice popisuje frekvenci výskytů slov v dokumentech. Její velikost je $M \times N$, kde M je počet unikátních slov ze všech dokumentů a N je počet dokumentů. Hodnota na pozici $[i, j]$ udává, kolikrát se slovo w_i vyskytlo v dokumentu D_j. Jednotlivé řádky matice znázorňují vektory slov a sloupce vektory dokumentů.[3]

Běžně jsou tyto matice velice řídké, tudíž informace mohou být koncentrovány do menšího počtu dimenzí. Toho je dosaženo přes metodu SVD, kde se provádí zobrazení každého slova do n-dimenzionálního podprostoru, který dává nejlepší aproximaci (metodou nejmenších čtverců) na originální data. Takto reprezentujeme každé slovo při využití n nejvýznamnějších 'latentních proměnných', a proto se tento proces nazývá latentní sémantická analýza (LSA).[3]

V praxi se také využívá matice Document-Term, která je transponovaná T-D, tudíž jsou na řádcích dokumenty a slova ve sloupcích.

3.3.2 TF-IDF

Tato matice je obdoba T-D, má stejné parametry (řádky - slova, sloupce - dokumenty), avšak znázorňuje relevanci slov v dokumentech. Následující rovnice určují jaké bude mít matice hodnoty. Výsledná hodnota nám ohodnotí slovo v dokumentu v závislosti na všech dokumentech, získáme tak přehled o důležitých slovech a naopak slova, která se objevují v dokumentech často budou zanedbána.[4]

- TF - Relativní četnost termu v dokumentu. [4]

$$tf_{t,d} = \frac{n_{t,d}}{\sum_{k \in d} n_{k,d}}$$

t ... slovo

d ... dokument

k ... slovo v dokumentu

n ... četnost slova

- IDF - Převrácená četnost slova ve všech dokumentech. Tato složka nám určí, jak moc je slovo časté, když se tedy vyskytuje ve většině dokumentů, bude výsledné *idf* velice malé.[4]

$$idf_{t,D} = \log \frac{|D|}{|\{d \in D \ : \ t \in d\}|}$$

D ... všechny dokumenty

- Výsledná hodnota se vypočítá součinem předchozích dvou složek:[4]

$$tfidf_{t,d} = tf_{t,d} \times idf_{t,D}$$

3.3.3 Content bearing words

Content bearing words (CBW) je varianta LSA, kterou vynalezl Hinrich Scütze (1998) za účelem měření sémantické podobnosti mezi slovy. Oproti předchozím maticím typu T-D, které používají dokumenty pro označení sloupců, je tato matice symetrická ($N \times N$) a používá v obou dimenzích unikátní slova.[3]

Matice obsahuje tzv. *content-bearing* slova neboli vektory, které obsahují informaci o tom, která jiná slova se vyskytla v rámci jejich kontextu. Hodnota pro slovo se zvyšuje, pokud se objeví v tzv. kontextovém okně těchto *content-bearing* slov. Kontextové okno velikosti $x \in \mathbb{N}$ si můžeme představit jako okolí kolem slova, přičemž se myslí x slova za a x slov před vybraným slovem. Celková velikost kontextového okna (CW) je tady maximálně $2x$. Proto například slovo *'fotbal'* je určováno faktem, že se často vyskytuje poblíž slov jako *'sport'*, *'zápas'* atd.[3]

Tato metoda má velmi dobré výsledky při zkoumání sémantiky slov (např. shlukování slov stejného významu, hledání homonym). Vektorový prostor, který je takto vytvořen a jehož body jsou zde reprezentovány jako slova a pojmy, se také nazývá *WORD-SPACE*.[3]

3.3.4 Matice podobnosti

Matice podobnosti (SM) slouží pouze jako jedna z forem experimentu. Jedná se o symetrickou matici velmi podobnou *Content bearing words*, tedy řádky i sloupce matice reprezentují slova. Hodnoty jsou vypočítány z jakékoliv již zmíněné matice M, kde pro pozici i, j spočítáme míru podobnost mezi vektory $w_i \in M$ a $w_j \in M$. Po normalizaci mají vektory stejný tvar jako v rovnici kvantového stavu, kde každá hodnota vektoru udává pravděpodobnost s jakou se může nahradit jiným slovem. Velká nevýhoda této matice je její velká časová složitost při vytváření.

$$N \times M : O(N^2 M) \rightarrow N \times N : O(N^3)$$

Vektory slov znázorňují kvantový stav, například:

$$|jablko\rangle = \sqrt{\frac{4}{5}}|ovoce\rangle + \sqrt{\frac{1}{10}}|hruška\rangle + 0|budova\rangle + \dots$$

Tento vzorec lze vyložit tak, že slovo *jablko* má nejvíce společného se slovem *ovoce*, neboť náleží do této velké množiny a z tohoto důvodu si je méně podobné se slovem *hruška*, ale naopak nemá nic společného se slovem *budova*. Takto lze vyjádřit každé slovo v této matici.

3.4 Dotazy

Tato část popisuje tvorbu dotazu, který bude ve formě vektoru (popř. vektorového podprostoru).

Nejdříve definujeme operaci negace za pomoci ortogonality, a disjunkci pomocí součtu vektorových podprostorů.[3]

3.4.1 Negace vektoru

Potřebujeme vytvořit dotaz tak, aby například výraz *apple* NOT $fruit$ systém pochopil, že se zajímáme o počítačovou společnost, nikoliv o ovoce. To zahrnuje nalezení významových rozdílů slova *apple*, které nesouvisejí se slovem *fruit*. Významy těchto slov spolu nesouvisejí právě, když nemají žádné společné složky, stejně jako dokument je zcela nerelevantní, pokud je jeho skalární součin s dotazem uživatele roven nule - přesněji, když jsou dotazovaný vektor a vektor dokumentu na sebe kolmé (ortogonální). Definice negace pro vektory se opírá o tuto shodu mezi nerelevantností a ortogonalitu v prostoru slov (*WORD-SPACE*).[3]

Definice 1 *Dvě slova a a b jsou mezi sebou považovány za nerelevantní, pokud jsou jejich vektory kolmé, tj. a a b jsou vzájemně nerelevantní, pokud $a \cdot b = 0$.* [3]

Výraz 'a NOT b' je nyní interpretován jako 'ty složky a, které jsou nerelevantní pro b'.

Definice 2 *Nechť V je vektorový prostor se skalárním součinem. Pro vektorový podprostor $A \subseteq V$ definujeme ortogonální podprostor $A^{\perp}$ jako podprostor* [3]

$$A^{\perp} \equiv \{v \in V : \forall a \in A, a \cdot v = 0\}.$$

Nechť A a B jsou podprostory V. Pro NOT B je myšleno $B^{\perp}$ a pro A NOT B je myšleno zobrazení A na $B^{\perp}$. [3]

Nechť $a, b \in V$. Pro a NOT b je myšleno zobrazení a na $\langle b\rangle^{\perp}$, kde $\langle b\rangle$ je podprostor [3]

$$\{\lambda b : \lambda \in \mathbb{R}\}.$$

Nyní můžeme předvést jednoduché výpočty s jednotlivými vektory z prostoru slov.

Teorém 1 *Nechť $a, b \in V$. Potom a NOT b je reprezentováno vektorem*

$$a \text{ NOT } b \equiv a - \frac{a \cdot b}{|b|^2} b$$

kde $|b|^2 = b \cdot b$ je norma b. [3]

Z toho vyplývá, že vektor a NOT b je přesně ta část vektoru a, která je nerelevantní s b (podle Definice 1) tak, jak jsme požadovali.

Pro normalizované vektory lze tuto rovnici zjednodušit do tvaru

$$a \text{ NOT } b = a - (a \cdot b)b.$$

V praxi je tento vektor znovu normalizován. Tento proces na výpočet dotazu je velmi efektivní. Pokud chceme najít termíny nebo dokumenty, které jsou relevantní k dotazu a NOT b, musíme porovnat všechny kandidáty s a a také b, a potom vypočíst jejich rozdíl. Teorém 1 nám dává jeden vektor pro takový dotaz, a proto, pokud chceme najít podobnost s jakýmkoliv vektorem, musíme mezi nimi provést pouze skalární součin. [3]

3.4.2 Vektorová disjunkce a konjunkce

Modelování disjunkce výrazu (např. A OR B) funguje podobně jako v předchozím případě. Disjunkce je v teorii množin definována jako sjednocení ($\cup$), což v lineární algebře odpovídá součtu podprostorů, jelikož $A + B$ je nejmenší podprostor V, který obsahuje současně A i B. [3]

Definice 3 *Nechť $b_1 \ldots b_n \in V$. Výraz b_1 OR ... OR b_n je reprezentován jako podprostor* [3]

$$B = \{\lambda_1 b_1 + \ldots + \lambda_n b_n : \lambda_i \in \mathbb{R}\}.$$

Najít podobnost mezi jedním slovem a a celým podprostorem B je složitější než najít podobnost mezi jednotlivými slovy. Lze proto definovat

$$sim(a, B) = a \cdot P_B(a),$$

jde o skalární součin a s $P_B(a)$ (zobrazením a na podprostor B), protože takto dostaneme velikost vektoru, který leží v podprostoru B. Aby byla tato podobnost nalezena v praxi, nelze počítat $sim(a, b_j)$ pro každý vektor b_j z podprostoru B zvlášť. Vektory musí být ortonormální, tj. jsou párově ortogonální a mají jednotkovou velikost. Proto je nutno sestrojit ortonormální bázi pro B. V praxi je možno využít například Gram-Schmidtova procesu, čímž je získána ortonormální báze $\{\tilde{b}_j\}$ pro podprostor B. Poté můžeme použít vzorec

$$P_B(a) = \sum_j (a \cdot \tilde{b}_j)\tilde{b}_j,$$

z čehož vyplývá vzorec pro podobnost

$$sim(a, B) = \sum_j (a \cdot \tilde{b}_j).$$

Při výpočtu této podobnosti je sčítán skalární součin a s každým vektorem $\tilde{b}_j$. [3]

Tímto je, ale ztracena metoda negace z teorému 1. Pokud mají být tyto dvě metody spojeny, a tedy negovány více než jedním slovem najednou, je nutno nalézt novou metodu.

Pokud tedy chceme dokument a, ale nechceme dokumenty $b_1, b_2, \ldots, b_n$, je jasné, že chceme pouze dokumenty, které nejsou relevantní pro všechny nežádoucí b_j. Hledáme tedy výraz,

$$a \text{ AND } (\text{ NOT } b_1) \text{ AND } (\text{ NOT } b_2) \ldots \text{ AND } (\text{ NOT } b_n)$$

který lze převést do formy

$$a \text{ NOT } (b_1 \text{ OR } \ldots \text{ OR } b_n).$$

S využitím definice 3 můžeme namodelovat disjunkci b_1 OR $\ldots$ OR b_n jako podprostor

$$B = \{\lambda_1 b_1 + \ldots + \lambda_n b_n : \lambda_i \in \mathbb{R}\}$$

a tento výraz můžeme přiřadit jednomu vektoru, který je ortogonální na všechny nevyžádané vektory $\{b_j\}$. Vektor pro tento výraz lze tedy vyjádřit takto

$$a \text{ NOT } (b_1 \text{ OR } \ldots \text{ OR } b_n) \equiv a - P_B(a),$$

kde P_B je zobrazení na podprostor B jako u rovnice podobnosti z definice 3. Proto stačí jediný skalární součin a získáme tak vylepšenou možnost zadávat složitější dotazy.[3]

3.5 Metrika a norma

V předchozích kapitolách byly popsány různé matice pro slova. V experimentech budou využívány jako vektory slov.

3.5.1 Kosinova podobnost

Jedná se o míru podobnosti mezi dvěma nenulovými vektory v Eukleidovském prostoru o N dimenzích, která měří kosinus úhlu mezi nimi. Kosinus 0°je 1 a pro interval <0;2π) je menší než 1. Proto vektory, které jsou stejné mají podobnost 1 a ty, které jsou mezi sebou v úhlu 90°, mají podobnost 0. Pokud jsou vektory diametrálně opačné je jejich podobnost -1. Jelikož míra nebere v potaz velikost vektorů, lze porovnávat i velikostně velmi odlišné vektory.[3]

$$sim(a, b) = \frac{a \cdot b}{\|a\| \, \|b\|} = \frac{\sum a_i b_i}{\sqrt{\sum a_i^2 \sum b_i^2}}$$

3.5.2 Eukleidovská norma

Norma je funkce, která každému nenulovému vektoru přiřazuje kladné reálné číslo (tzv. délku nebo velikost), nulový vektor jako jediný má délku 0. Existuje několik druhů norem, ale ve svých výpočtech budu používat Eukleidovskou z důvodu zjednodušení několika výpočtů a také proto, že má podobné vlastnosti jako kvantový stav.

Eukleidovská norma vektoru $\vec{v} = (v_1, v_2, \ldots, v_n)$ je definována na prostoru $\mathbb{R}^n$ jako:

$$\|v\| := \sqrt{v_1^2 + \ldots + v_n^2}$$

Pomocí této normy jsou vektory normalizovány. Normalizace se používá, pokud jsou data rozdílných velikostí. I když jsou si vektory podobné, je výhodné je normalizovat, jelikož se mohou zjednodušit některé výpočty. Normalizovaný vektor podle Eukleidovské normy obsahuje čísla v intervalu <0;1>. V matici jsou rozlišovány sloupcové a řádkové vektory, ale postup normalizace je pro oba typy stejný.

Normalizovaný vektor $\vec{n}$ z vektoru $\vec{v}$ je definován takto:

$$\vec{n} = \frac{\vec{v}}{\|v\|}$$

4 Experimenty

4.1 Vytvoření seznamu unikátních slov

Jedná se o proces, při kterém se zpracovává text přirozeného jazyka, rozděluje ho na slova, která poté filtruje a ve výsledku vytváří seznam unikátních slov.

4.1.1 Rozdělení slov

Prvně budou načítány dokumenty do paměti. V každém dokumentu bude procházen jeho text, který bude nejdříve rozdělen na slova podle mezer. Jelikož slova nebudou vždy v čisté podobě, mohou obsahovat různé nežádoucí znaky (např. tečka, čárka, závorky atd.), které je potřeba odstranit. Tyto znaky byly odstraněny pouze ze začátku a konce slova. Poté se vybírají různá sousloví se spojovníkem. Ty jsou rozlišeny podle toho, jestli se v nich vyskytují znaky '-', '/' nebo '\' a podle těchto znaků jsou rozděleny na dvě slova. Následující pravidlo je poněkud speciální a dá se použit výhradně pro angličtinu. Jde o odstranění přivlastňovací přípony *s* neboli *'s*. Jelikož následně byl použit stematizační algoritmus, může se poslední krok zdát redundantní, ale dle autorových zkušeností, použitý stemmer nebyl bezchybný a v některých případech nedošlo k odstranění tohoto suffixu. Nakonec jsou slova převedena na malá písmena a jsou kontrolovány, zda neobsahují nežádoucí symboly (pouze písmena). Slova, která nevyhovují, jsou zahozena, jelikož u nich není jistota, že se jedná o slova a ne například o číslovky. Toto je základní úprava a takto odfiltrovaná slova mohou vytvořit unikátní slovník, další kroky jsou pouze volitelné. Většinou to však nestačí, jelikož takto vytvořený slovník obsahuje mnoho neplnovýznamových slov, které působí jako šum a také obsahuje plno redundancí v podobě více tvarů jednoho slova. Postup základního rozdělení dokumentů na slova:

1. Převod textu na slova

   např. *Hello world!* → {*Hello, world!*}

2. Odstranění jiných znaků než písmen z afixů

   např. *"hello!"* → *hello*

3. Rozdělení sousloví se spojovníkem

   např. *sugar-free* → {*sugar, free*}

4. Odstranění přivlastňovací přípony *s*

   např. *google's* → *google*

5. Převod velkých písmen na malá ([*'A'–'Z'*] → [*'a'–'z'*])

   např. *HOME* → *home*

4.1.2 Použití negativního slovníku

Po předchozím kroku seznam unikátních slov obsahuje mnoho neplnovýznamových termínů, které mohou negativně ovlivnit výsledky vyhledávání. Proto je potřeba množství těchto slov co nejvíce redukovat.

Stoplist obsahuje dvě metody filtrace:

1. Volba krátkých slov - slova, která jsou příliš krátká mají velkou pravděpodobnost, že se jedná o stopslovo, nebo že toto slovo nepřináší mnoho informací o jeho významu, a proto je příliš obecné. Pro potřeby filtrování byly filtrovány slova délky dvě a kratší.

2. Volba druhů slov, které nenesou význam a slouží pro syntaktické účely - byl použit předefinovaný seznam spojek, předložek, zájmen atd., který obsahuje 555 slov. Ukázka stoplistu:[4]

   a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with, ...

   Celý seznam je k nalezení v příloze na CD.

Jedna z nevýhod jsou krátká slova, která mají nějaký význam, nejčastěji zkratky (např. *'PC'*, *'IT'*). V tom případě by měl být použit anti-negativní slovník s těmito slovy, avšak tato část není zcela nutná.

4.1.3 Použití stematizačního algoritmu

Stematizace pomáhá redukovat počet unikátních slov, tím, že je převádí na základní tvar (kmen nebo kořen). Byl použit algoritmus *Porter Stemmer* zaměřený na angličtinu. Nebyl problém s jeho použitím v programu, protože jeho implementace je v mnoha programovacích jazycích (např. C, C#, Java atd.). Jeho kód není příliš rozsáhlý, jelikož neobsahuje slovník slov, ale pouze pravidla, která zaměňují afixy slov. Kdyby takový slovník používal, jednalo by se o lematizátor.

Algoritmus rozhodně není dokonalý, ale odvede dobrou práci při spojování stejných slov dohromady. Algoritmus má tendenci nechávat apostrofy u přivlastňovací přípony, např. *company's* → *company'*. To je způsobeno nesprávným použitím jednoduchých uvozovek namísto apostrofy, což jsou odlišné znaky. Tento problém byl vyřešen v základní filtraci (kapitola 4.1.1).

Mezi další problémy patří dosazování nesprávných přípon. Většinou se jednalo o koncovky *e*, které úplně odebral (např. *apple* → *appl*) a *y*, který měnil na *i* (např. *company* → *compani*). Jelikož tento problém byl spíše po vizuální stránce, byl pouze použit seznam, který obsahoval původní podoby slova, (např. slovo *compani* bylo odvozeno ze slov *company* a *companies*).

Pokud by byl tento algoritmus použit, musí být také provedena úprava textů z dokumentů. Nejlepší řešení je, aby každý dokument měnil nebo odstraňoval tato slova už rovnou při jeho prvním čtení.

4.2 Převod do matic

Po získání seznamů dokumentů a unikátních slov se s ním může pracovat dále. Převod na vektory byl proveden pro tyto matice:

- Term-Document

  Při procházení seznamu dokumentů se zvyšuje hodnota v matici daného slova při jeho výskytu.

- TF-IDF

  Matice může být vytvořena buď z T-D matice nebo procházením dokumentů. První možnost velice usnadní výpočty, jelikož frekvence, které jsou potřeba, jsou částečně spočteny. Stačí dopočítat celkový počet dokumentů a pro každou hodnotu pak celkový počet slov v dokumentu. Poté už stačí dosadit do vzorců, které jsou popsány v kapitole 3.3.2.

- Content Bearing Words

  Oproti předchozím maticím, kde byl možný pouze jeden výsledek, zde je nutno zvolit velikost kontextového okna. Jeho velikost také určuje, kolik informací bude obsahovat vektor slova. Pokud bude zvoleno příliš velké okno, bude vektor velmi obecný, pokud moc malé, nebude obsahovat informace, které by jinak byly očekávány. Pro experimenty byly zvoleny tyto velikosti: 5, 15, 30.

- Matice podobnosti

  Tato matice je tvořena z jiné již vytvořené. Není proto třeba procházet dokumenty. Velká nevýhoda SM je její až kubická časová složitost, která však lze zmírnit, když využijeme její symetrických vlastností.

  Pro experimenty byla použita matice *Content Bearing Words* jako vzorová. Její výpočet byl velmi zdlouhavý i paměťově náročný.

Po převodu musí byt matice normalizovány podle řádků, tedy pro vektory slov.

4.3 Tvorba dotazů

Aby bylo možné systému vyhledávat co nejefektivněji, je potřeba vytvářet dotazy, které ponesou co nejvíce informací na požadované téma. Dotazem v systému se rozumí vektor, který je vytvořen ze zadaného výrazu. Celý výraz se skládá vždy z určitých slov, která systém obsahuje. Bohužel není možné vytvářet nová slova, pouze je napodobit výsledným dotazem.

Ve tvorbě dotazů byla využita kvantovou logiku, která byla detailně popsána v kapitole 3.4. Ta bude poté rozšířena, aby bylo možné zadávat ještě komplexnější dotazy.

V systému existuje několik typů dotazu:

1. Samotný vektor slova

   Vektor není nijak upraven, pouze vyjmut z matice.

2. Několik slov najednou

   Vektory jsou opět vyjmuty z matice bez jakýchkoliv úprav. Jde ovšem o celou množinu těchto vektorů, proto je vyhledávání složitější. Tento dotaz je vhodný, pokud jsou vyžadovány společné výsledky pro dvě a více slov. Obecný tvar dotazu:

   $$(a_1 \text{ OR } \ldots \text{ OR } a_n)$$

3. Negace vektoru slova jiným slovem

   Pokud je například zadán výraz *apple* NOT *computer*, požadujeme výsledky související s ovocem a naopak jsou nežádoucí výsledky, které by souvisely s počítačovou firmou. Je potřeba, aby vektor pro slovo *apple* byl kolmý na vektor slova *computer*. Toho lze docílit tak, že bude použita operace NOT definovaná v kapitole 3.4.1. V obecném tvaru dotaz vypadá takto:

   $$a \text{ NOT } b$$

4. Rozšířená negace slova s vektory více slov

   Je-li potřeba předchozí výraz více zpřesnit, například tak, aby výsledky neobsahovaly záznamy o jablkovém koláči, je potřeba provést negaci na oba vektory najednou. Pro zadání více vektorů naráz bude použita operace disjunkce, která se zapisuje slovem OR . Ovšem není možné použit stejný vzorec jako v předchozím případě, kde byl pouze vytvořen ortogonální vektor. Tentokrát je potřeba, aby dotazový vektor byl kolmý na celý podprostor našich nežádoucích vektorů. Potom je možné zadat výraz jako *apple* NOT (*computer* OR *pie*). Celý postup této operace je popsán v kapitole 3.4.2. Obecný tvar dotazu:

   $$a \text{ NOT } (b_1 \text{ OR } \ldots \text{ OR } b_n)$$

5. Negace více slov s jiným slovem

   Tento dotaz je pouze kombinací typu 2 a 3. Je zde negace po jednom slovu a poté procházena celá množina takto negovaných slov.

   $$(a_1 \text{ OR } \ldots \text{ OR } a_n) \text{ NOT } b$$

6. Negace více slov s vektory více slov

Jde opět o variantu, která kombinuje typ 2 a 4. Jde o nejvíce komplexní dotaz, jaký systém umožňuje.

$$(a_1 \text{ OR } \ldots \text{ OR } a_m) \text{ NOT } (b_1 \text{ OR } \ldots \text{ OR } b_n)$$

4.4 Hledání dotazů

Hlavní účel systému je vyhledávání komplexních dotazů pomocí kvantové rovnice. První část byla zaměřena na podobnost se slovy, hlavním účelem bylo prozkoumat, jak funguje operace negace (NOT) a jak efektivně dokáže odfiltrovat nežádoucí slova.

Další část se věnuje podobnosti s dokumenty. Porovnává se tedy slovo (nebo výraz) s dokumentem. Nejedná se tedy o porovnávání vektorů dokumentů, které by byly získány z matic typu T-D a TF-IDF. Nejprve bylo vyzkoušeno kolik informací v sobě nesou vektory z matic T-D a TF-IDF, když jsou vyjmuty pouze vektory slov. Poté byla část zaměřená na převedení dokumentu do matice, která byla porovnávání jako vektor s podprostorem vektorů. Druhá metoda už byla možná provést pro všechny typy matic.

4.4.1 Podobnost se slovy

Pro typy dotazů, které reprezentuje pouze jeden vektor (typy 1, 3 a 4) se provádí jednoduché porovnávání s ostatními. Pro složitější dotazy, které obsahují množinu vektorů (typy 2, 5, 6), je nutno provést porovnání pro každý vektor z množiny dotazu s každým vektorem slova v matici, kde jsou poté uloženy nejlepší (nejvyšší) výsledky pro stejná slova.

Vzájemnou míru podobnosti při porovnávání vektorů je určena kosinovou podobností (kapitola 3.5.1).

V experimentech jsou použity všechny matice, avšak hlavní účel těchto testů byl pro matici CBW. Následně bylo zjišťováno, jakou má přesnost SM ve srovnání s ostatními.

Poslední, na co byla tato část zaměřena bylo, jaký vliv má velikost kontextového okna v matici CBW na výsledky. Experimenty byly provedeny se třemi velikostmi (5, 15, 30).

4.4.2 Podobnost s dokumenty

Zkoumání podobnosti vektorů s celými dokumenty je další část výzkumu této práce. Tato část by měla interpretovat autorovo chápání kvantové logiky jako celku, jelikož hledané slovo je samo o sobě množinou významů a proto je nesprávné ho porovnávat pouze s jedním slovem, ale s celou skupinou slov, neboli dokumentem jako celkem.

Nejprve bylo zjišťováno, jak dobře lze určit podobnost dokumentu s vektorem na základě matic T-D a TF-IDF. Jelikož tyto matice obsahují samotné vektory dokumentů ve sloupcích, řádkový vektor byly interpretován jako slovo a jednotlivé složky vektoru jako pravděpodobnosti pro dané dokumenty. Takto byl napodoben kvantový stav pro každé slovo. Poté bylo možno provést operace negace a disjunkce jako s každým dotazem.

V druhé části byl dokument přizpůsoben tak, aby vyhovoval prostoru slov. Úprava dokumentu probíhá tak, že každé unikátní slovo obsažené v dokumentu bylo nalezeno v jedné z již vytvořených matic a uloženo do nové matice. Takto vznikne nová matice, která je vektorový podprostor. Z této matice dokumentu je potom možné získat ortonormální bázi, která je připravena k porovnání s dotazy. Tento proces je popsán detailně v kapitole 3.4.2. Výsledná podobnost je počítána přes tento vzorec:

$$sim(a, B) = a \cdot P_B(a)$$

Tento experiment je možné provést pro všechny typy matic, avšak pro experimenty byly použity pouze matice CBW a SM, jelikož nejlépe reprezentují význam slova.

5 Testovací systém

Aby mohly všechny poznatky být ověřeny v praxi, bylo nutné vytvořit systém, který umožní načítat dokumenty a dotazovat se na ně. Jelikož nebyla nalezena žádná veřejně dostupná implementace systému, která by obsahovala všechny požadované matice a operace, bylo nutné tento systém vytvořit. Pro tyto účely byla vytvořena konzolová aplikace, která obsahuje vlastnoručně napsanou knihovnu s operacemi pro vytváření dotazů a jejich vyhledávání. Pro maticové operace byla využita knihovna *Math.NET*, jež tuto práci velmi usnadnila.

Jelikož hlavní účel této práce nebyl v implementaci takového systému, ale ve zkoumání již popsaných metod, bude jeho popis stručný a nebude obsahovat přesné programátorské techniky při jeho implementaci. Bude pouze popsána jeho strukturu a práce s ním.

5.1 Knihovna Math.NET

Tato knihovna společnosti *Numberics* obsahuje metody z několika odvětví matematiky, pro účely této práce však stačily ty z lineární algebry. Byly využity především metody pro skalární součiny vektorů a matic a poté jejich normalizaci. Velmi jsem ocenil funkci pro výpočet ortogonální báze, kterou bych jinak musel složitě implementovat v mém programu. Mimo tyto operace knihovna umožňuje také výpočet SVD, který je součástí modelu LSA. Jak název napovídá, její implementace je v programovacím jazyce C#. [7]

5.2 Implementace

Pro vlastní implementaci se nabízelo několik programovacích jazyků. Pro velkou rozličnost matematických operací by byl vhodný *MatLab*, naopak pro optimalizaci výkonu programu by byl ideální C++. Pro systému byl však zvolen jazyk C#, jelikož pro něj existuje velké množství knihoven a autor ním má největší zkušenosti. Systém byl naprogramován ve vývojovém prostředí *Visual Studio 2015* od společnosti *Microsoft*.

Byla zvolena 3-vrstvou architekturu, pro její přehlednost a účelnost.

- Datová vrstva (*DataStorage*)

  Vrstva obsahuje třídu *Storage*, která se chová jako databáze. Tato načítá, ukládá a filtruje všechny dokumenty a termíny, které do systému vložíme. Také obsahuje třídy entit, které jsou *Document* a *Term*. Pro filtrování jsou zde třídy *StopList*, která obsahuje negativní slovník a dále třída *PorterStemmer*, kde je naimplementovaný algoritmus pro stematizaci. Pouze tato vrstva pracuje s těmito daty a přes funkce je předává do další vrstvy.

- Doménová vrstva (*Logic*)

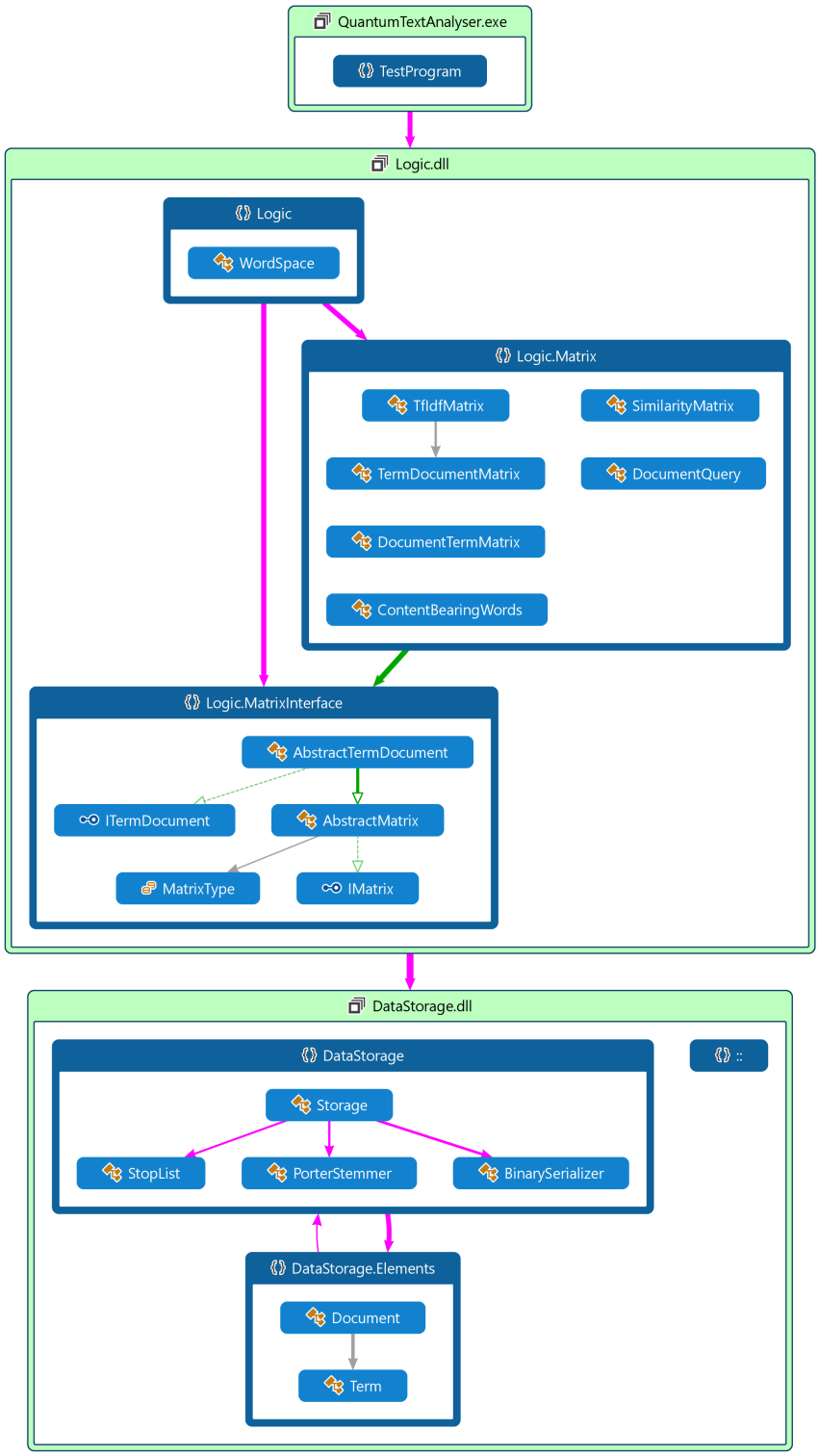
  Tato vrstva obsahuje hlavní třídu *WordSpace*, která používá všechny typy matic a slouží jako prostředek pro naše experimenty.

Matice jsou rozděleny tak, že dědí z abstraktní třídy (*AbstractMatrix*, *AbstractTermDocument*), které implementují rozhraní (*IMatrix*, *ITermDocument*). Většina operací se proto nachází v abstraktní třídě *AbstractMatrix*. Každá konkrétní třída matice implementuje svou vlastní metodu načítání dat, jak bylo popsáno v kapitolách 3.3 a 4.2.

- Prezentační vrstva (*TestProgram*)

  Tato vrstva obsahuje pouze konzolovou aplikaci (třídá *TestProgram*), která používá třídu *WordSpace* z předchozí vrstvy. Provádí se na ní dotazy a výpisy výsledků.

Datová a doménová vrstva dohromady slouží jako knihovna a v kombinaci s *Math.NET* ji lze použit na jakémkoliv operačním systému podporující *.NET framework 4.5*.



Obrázek 1: Třídní diagram systému

6 Výsledky

Z výsledků by měly být zřejmé očividné výhody a nevýhody kvantové logiky ve vyhledávání ať už slov nebo celých dokumentů.

6.1 Testovací data

Pro testování byly použity články z webových stránek *Wikipedia.org* z roku 2017. Články nebyly autorem vlastnoručně zpracovávány, ale získány již připravené ze stránek *corpusdata.org*[8], kde jsou k dispozici textová data pro analýzu textu. Data, získaná z webových stránek *Wikipedia*, byla zvolena pro jejich přesný popis pojmů. Datová sada obsahuje 2913 článků různých délek, na různá témata. Pro účely zkoumání bylo vybráno sto z nich.

Část článků (62) byla zvolena tak, aby se v jejich textu vyskytovala zadaná slova, popřípadě slova opačná, která by mohla negativně ovlivnit hledání. Slova, na která byla zaměřena pozornost byla:

- *Apple* - druh ovoce nebo počítačová firma.

  Slovo je homograf, proto je ideální na vyzkoušení výrazu negace.

- *Google* - počítačová firma

  Počítačová firma, která byla zvolena pro zajištění více členů z jedné významové skupiny.

- *Microsoft* - počítačová firma

  Počítačová firma, která byla zvolena pro zajištění více členů z jedné významové skupiny.

- *Juice* - ovocný nápoj

  Slovo zvolené pro zajištění, že se ve vyhledávání objeví více zástupců ovoce.

- *Beatles* - hudební skupiny

  Slovo z kategorie, která přímo nesouvisí s předchozími vybranými slovy.

Zbylých 38 článků bylo vybráno náhodně, aby nedošlo k uměle vytvořeným pozitivním výsledkům. Tyto dokumenty by měly působit jako šum a napodobit tak reálný svět, kde podmínky nejsou vždy ideální.

6.2 Vliv filtrů

Celková lexikální analýza na výsledky vyhledávání i na paměťovou náročnost programu. Pro testování byla zvolena dvě kritéria:

Tabulka 1: Testování filtrovacích metod pro texty *Wikipedia*

Počet dokumentů		Základní	SL	PS	Oba filtry
100	Počet slov	75 065	33 853	75 065	33 453
100	Unikátní slova	10 296	9 770	7 189	6 770
2913	Počet slov	1 549 860	722 932	1 549 860	714 699
2913	Unikátní slova	55 332	54 413	39 112	38 326

1. Celkový počet slov

   Toto kritérium vyhodnotí jaké množství textu je k dispozici pro další zpracování.

2. Počet unikátních slov

   Díky tomuto počtu je možné zjistit jak velké budou matice, což významně ovlivňuje dobu jejich výpočtu.

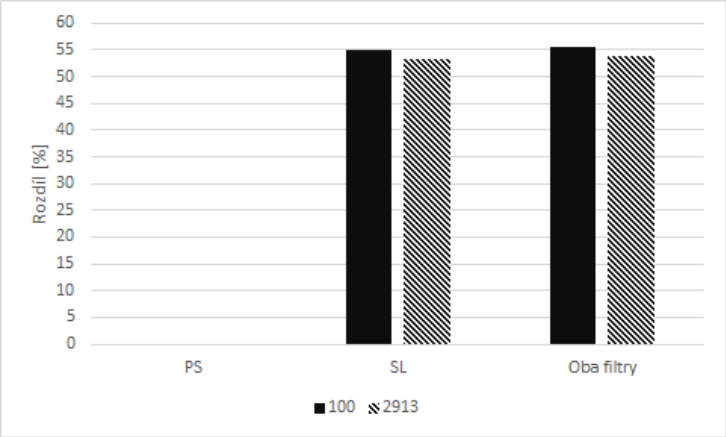
Filtrování slov v lexikální analýze probíhá ve třech fázích. První fází je rozdělení slov, která se provádí vždy. Druhou fází je použití tzv. stoplistu (negativního slovníku) a poslední, třetí, fází je použití algoritmu pro stematizaci. Tabulka 1 udává výsledky pro jednotlivá nastavení filtrování slov. Bylo použito sto článků, vybraných autorem, a pro srovnání bylo zároveň využito všech textů datové sady (celkem 2913). Stoplist znázorněný v tabulce 1 obsahuje 555 slov.

Lze si povšimnout, že před filtrací databáze obsahovala velké množství slov, které by bylo nutné zdlouhavě zpracovávat. Seznam unikátních slov obsahuje mnoho slov neplnovýznamových, například *the* nebo *at*, která stěžují vyhledávání.

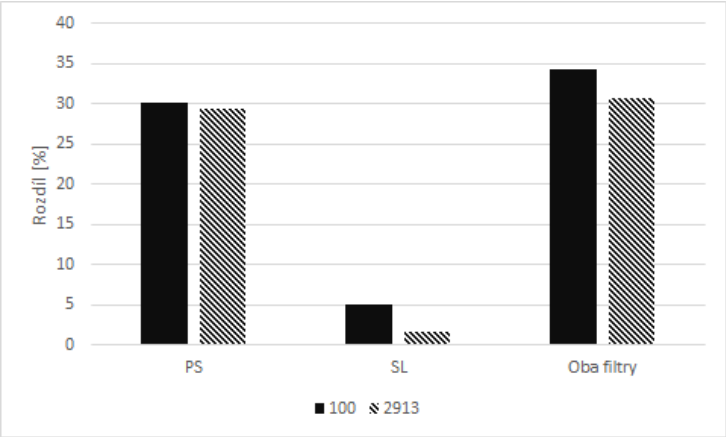
Použitím stoplistu bylo množství prohledávaného textu zredukováno. Zároveň bylo zjištěno, že neplnovýznamová slova zabírají více než polovinu textu (viz Obrázek 2). Oproti tomu počet unikátních slov klesl jen minimálně (viz Obrázek 3). Srovnání vlivu při použití velkého počtu dokumentů dokázalo, že efektivita redukce unikátních slov klesá a to v nepřímé úměře k počtu dokumentů.To je způsobeno tím, že stoplist má konstantní počet slov a jeho jediná dynamická složka tkví v odstraňování slov kratších než 3 znaky, která nejsou příliš častá. Redukce celkového počtu slov je téměř konstantní a v obou případech se pohybuje okolo 50%.

Na druhou stranu, stematizační algoritmus upravuje data z jiného hlediska. Velikost textu zůstává zachována, ale významně se snižuje počet unikátních slov (viz Obrázek 2). Je možno hovořit o rozdílu okolo 30% oproti původnímu počtu, což lze vidět na grafu z obrázku 3. Opět si lze všimnout, že je tento rozdíl konstantní a to bez ohledu na množství dokumentů. V mnoha případech však není tato redukce ideální a jde o znehodnocování slov. Použití tohoto filtru při hledání významu slov je dle mého názoru spekulativní, neboť má za následky zobecňování významů slov, což může být nežádoucí. Na druhou stranu, stematizační algoritmus odstraňuje mnoho redundantních významů pro slova v různých časech a podobné změny tvaru slova.

Při použití obou filtrů, stoplistu i stematizačního algoritmu, lze vidět stejné chování jako v jednotlivých případech. Dochází tedy k redukci textu a zároveň využívání pouze plnovýznamo-



Obrázek 2: Graf relativního rozdílu celkového počtu slov oproti základnímu pro 100 a 2913 dokumentů



Obrázek 3: Graf relativního rozdílu počtu unikátních slov oproti základnímu pro 100 a 2913 dokumentů

vých slov. V tabulce 1 si lze všimnout, že došlo k odstranění více unikátních slov, než když byly filtry použity zvlášť. Tento jev lze vysvětlit tak, že stoplist (SL) je použit ze všeho nejdříve, poté je na řadě Porter Stemmer(PS) a následně je slovo znovu prověřeno metodou SL. Takto lze odstranit případy, kdy neplnovýznamové slovo není v základním tvaru, a proto není obsaženo v SL, ale po jeho úpravě pomocí PS by bylo uloženo do databáze. Na základě vlastního pozorování bylo použito SL před PS z důvodu, že PS v některých případech neplnovýznamové slovo změní na špatný tvar, který není obsažen v SL a poté slovo uloží, čemuž je žádoucí se vyhnout.

6.2.1 Praktická ukázka

Čísla a počty slov málokdy vyjádří svou podstatu v praxi, a proto byla použita matice *Contetn Bearing Words*(CBW) s velikostí kontextového okna (CW) 15, aby bylo předvedeno několik praktických příkladů při vyhledávání.

Tabulka 2: Výsledky vyhledávání slova *company* bez použití SL

company		company NOT the	
company	1,000000	undervalued	0,447840
the	0,905027	reinvest	0,428892
value	0,904580	company	0,425354
currently	0,896180	restructure	0,399880
of	0,872324	growth	0,395088
measure	0,865979	value	0,388576
future	0,864290	earnings	0,387458
it	0,857921	eps	0,373523
investors	0,853156	dividend	0,366496
on	0,851324	measure	0,350628

Tabulka 3: Výsledky vyhledávání slova *strawberry* bez použití PS

strawberry		strawberry OR strawberries	
strawberry	1,000000	strawberries	1,000000
vodka	0,697217	strawberry	1,000000
peanut	0,677003	combier	0,971429
vanilla	0,677003	vodka	0,885714
flavoured	0,656167	ginger	0,857143
strawberries	0,597614	honey	0,678227
algae	0,592667	peanut	0,677003
combier	0,577694	vanilla	0,677003
pur	0,553912	flavoured	0,656167
aphrodisiac	0,553912	cucumber	0,655936

Pokud není použito SL, budou výsledky obsahovat mnoho neplnovýznamových slov (viz Tabulka 2). Pomocí operace NOT lze slova odstranit, avšak za cenu ztráty velkého množství správných výsledků. V případě výrazu *company* NOT *the* bude slovo *company* znehodnoceno natolik, že nebude uvedeno ani v nejlepších deseti výsledcích. To je způsobeno tím, že slovo *the* se objevuje v textu velmi často, a tudíž téměř každé slovo sdílí jeho významovou složku (viz Tabulka 5). Pokud je tedy použito v negaci, přijdeme i o mnoho relevantních slov. Pro vyhledávání je tedy téměř nutností SL použit.

Druhá tabulka znázorňuje vyhledávání bez použití metody PS. Hlavním problémem ve vyhledávání jsou redundantní výrazy. Tabulka 3 znázorňuje výsledky vyhledávání slova *strawberry*. Ve výsledcích vyhledávání se pak objevuje i množné číslo daného slova. Tuto nevýhodu lze částečně kompenzovat použitím výrazu OR . Výsledný výraz *strawberry* OR *strawberries* výsledky posílí, a lze v nich vidět, že se významově pohybuje okolo jídla a míchaných nápojů.

Ovšem pokud je použito této metody, může dojít ke spojování slov, která se budou poté těžce významově rozdělovat. Pokud se například použije slovo *job*, očekává se, že výsledky vyhledávání

Tabulka 4: Výsledky vyhledávání slov *job* a *jobs* bez použití PS

job		jobs		jobs NOT visionary	
job	1,000000	jobs	1,000000	jobs	0,639202
public	0,587806	tweaker	0,779241	fill	0,558637
claim	0,583169	inventor	0,775377	clarifying	0,528759
claimants	0,551468	visionary	0,769039	employers	0,459854
students	0,543252	gladwell	0,701791	employees	0,414318
schools	0,532022	innovation	0,681719	operations	0,383286
occupation	0,525958	repeatedly	0,565344	meetings	0,380909
education	0,522557	steve	0,528008	teachers	0,368879
struggling	0,503309	rightly	0,494170	custody	0,367056
taught	0,488811	tweak	0,486740	shipbuilding	0,344920

Tabulka 5: Výsledky vyhledávání slov *ministry* a *the* se základní filtrací

ministry		ministy NOT the		the	
ministry	1,000000	ministry	0,506562	a	1,000000
the	0,862204	broadcasting	0,386130	an	1,000000
government	0,836027	tewari	0,379167	the	1,000000
therefore	0,827971	relevance	0,366976	of	0,946006
minister	0,826349	satyajit	0,362261	past	0,924214
by	0,823229	outgoing	0,359568	by	0,923328
only	0,821702	redundant	0,358667	over	0,919056
tewari	0,820667	sushma	0,356232	new	0,915687
been	0,812675	ftii	0,348617	in	0,909766
past	0,809606	manish	0,337415	which	0,906337

budou souviset se zaměstnáním. Nicméně, množné číslo *jobs* by mohlo rovněž do svých výsledků vyhledávání zahrnout jméno zakladatele firmy Apple (Steve Jobs). Tabulka 4 tyto skutečnosti potvrzuje, jelikož se ve výsledcích objevují slova, která jsou úzce spjata s firmou Apple a její vůdčí osobností, například *invetor*, *Steve* či *visionary*. Třetí sloupec tabulky 4 ukazuje, že slovu *jobs* lze alespoň částečně jeho původní význam navrátit.

Pokud je tedy metoda PS použita, můžeme být vyhledávání specifických termínu ztíženo. I přesto tuto skutečnost byla v experimentech tato metoda použita.

Nejméně přesnou variantou je použití pouze základního filtrování. Nejenže budou výsledky velmi nepřesné, ale zároveň se velmi prodlouží čas vyhledávání. Výsledky vyhledávání jsou uvedenz v tabulce 5. Tabulka rovněž obsahuje výsledky větného členu *the*, která znázorňují jak mohou neplnovýznamová slova vypadat.

Tabulka 6: Výsledky vyhledávání slova *company* v CBW s velikostí CW 5, 15 a 30

company (CW=5)		company (CW=15)		company (CW=30)	
company	1,000000	company	1,000000	company	1,000000
valued	0,623449	holds	0,756452	cash	0,809165
undervalued	0,613057	stocks	0,752322	earnings	0,805512
growth	0,555614	sirius	0,748832	investors	0,803980
current	0,542658	cash	0,725765	holds	0,790483
shenanigans	0,531284	undervalued	0,700558	sirius	0,770892
nzi	0,527853	valued	0,694183	ratio	0,770385
ami	0,527853	earnings	0,685984	stocks	0,758456
lumley	0,526137	measure	0,682719	growth	0,757380
nutonomy	0,521276	growth	0,674016	measure	0,751397

6.3 Vliv velikosti kontextového okna v Content Bearing Words

I velikost kontextového okna v matici CBW může mít vliv na výsledky vyhledávání. Zároveň může mít i nepřímý vliv na matici podobnosti (SM), protože ta je z matice CBW tvořena. Pro zkoumání byly použity velikosti 5, 15 a 30. Součástí lexikální analýzy byly i metody SL a PS, ačkoli z tabulek (například tabulka 6) by mohlo vyplývat, že slova nejsou ve svém základním tvaru. Jedná se pouze o formu autorovy prezentace, kde jsou slova nahrazena prvním výrazem, který nebyl upraven stematizací (například *valued* je ve skutečnosti *value*). Autor se tímto snažil vyhnout nepřehledným výpisům, protože slova obsahovala chyby (například *compani* namísto *company*). Žádné další úpravy tabulky provedeny nebyly.

Zkoumáním bylo zjištěno, že velikost kontextového okna se ve výsledcích projevuje minimálně a její vliv je tudíž, téměř zanedbatelný. Vyjma některých případů, kde lehce převládala jedna velikost, byly výsledky téměř totožné. V návaznosti na toto zjištění je v práci uvedeno pouze několik příkladů.

Jako první příklad bylo zvoleno slovo *company*. V tabulce 6 lze pozorovat, že výsledky pro CW 15 a 30 se nijak zvlášť neliší a v obou případech je lze považovat za relevantní. Velikost 5 obsahuje méně přesné výsledky, nicméně jsou vynahrazeny menší podobností se slovy. Další experimenty tedy byly prováděny s velikostmi 15 a 30.

Další příklad využívá negaci. Na základě výsledků z tabulky 7 lze konstatovat, že lepší výsledky pro negaci nastaly v případě CW=15. V obou případech sice mají výsledky podobné významové skupiny, avšak v případě CW=30 se slovo *tree* nedostalo do nejlepších deseti výsledků, což ukazuje na to, že je více spojeno se slovem *christmas*, čehož se chceme vyvarovat.

Na základě předchozích dvou příkladů bylo v experimentech použito kontextové okno o velikosti 15.

Tabulka 7: Výsledky vyhledávání slova *tree* v CBW s velikostí CW 15 a 30

CW=15				CW=30			
tree		tree NOT christmas		tree		tree NOT christmas	
tree	1,00000	moss	0,64856	tree	1,00000	moss	0,71308
fundraising	0,80594	tree	0,60168	christmas	0,87514	bluebell	0,64102
christmas	0,79873	beards	0,55023	fundraising	0,86375	birds	0,63172
donation	0,79559	dragonflies	0,54780	donation	0,85744	perfectly	0,61346
cara	0,70811	birds	0,53672	cara	0,84906	fell	0,60704
guest	0,69247	lichen	0,53671	guest	0,83272	elder	0,59073
mckeough	0,66666	frosty	0,51701	bells	0,79149	coppiced	0,59008
bells	0,65493	elder	0,51094	cider	0,74907	limestone	0,59000
glenda	0,65162	coppiced	0,51094	mckeough	0,73116	clints	0,58328
cider	0,59031	hanging	0,47572	glenda	0,72199	bushes	0,58328

6.4 Hledání dotazů

Předchozí kapitoly určily, jaká data budou v následujících experimentech využívána. Jelikož už bylo zmíněno několik příkladů vyhledávání i s operací negace, lze navázat na složitější dotazy a srovnat rozdílné typy matic.

6.4.1 Podobnost se slovy

Tabulka 8 porovnává výsledky všech čtyř typů matic. Tyto matice lze rozdělit do dvou skupin:

1. Typ slovo-dokument
2. Typ slov-slovo

Tyto matice jsou na první pohled od sebe rozlišitelné. Pro vyhledávání bylo vybráno slovo *juice* a ve výsledcích se očekávaly různé ovocné příchutě a jiné nápoje nebo jídla.

Výsledky ukazují, že matice, které jsou spojeny přímo s jednotlivými dokumenty, jsou zaměřeny spíše na celý obsah článků než na samostatný význam slova. To jim ovšem neubírá na relevantnosti. Tyto matice mají nevýhodu při malém počtu dokumentů, jelikož i jejich vektory slov budou krátké. V praxi je ovšem opak pravdou, neboť dokumenty jsou tvořeny z často opakujících se slov, což znamená, že slov je omezený počet, ale dokumentů lze tvořit mnohonásobně více (běžně se používají až miliony dokumentů). Z tabulky 8 lze vyčíst, že z dvojice matice T-D a TF-IDF, má TF-IDF relativně nižší míru podobnosti s ostatními slovy, a proto má lepší výsledky matice T-D. TF-IDF má nevýhodu v tom, že degraduje velmi častá slova, což nemá vždy pozitivní vliv. Pokud se například v receptu často objevují ingredience, nemá význam jim dávat nižší váhu. I když by se mohlo zdát, že tento typ matice obohatí informace získané z dokumentů, dochází spíše k opaku. Z tohoto důvodu matice nebyla při srovnávání se slovy dále využívána.

Hlavním problémem tohoto typu matic je, že jsou příliš závislé na množství dokumentů. Pokud vyhledávané slovo obsahuje pouze jeden dokument, výsledky budou velmi zkreslené, neboť

Tabulka 8: Výsledky vyhledávání slova *juice* pro T-D, TF-IDF, CBW a SM

juice (T-D)		juice (TF-IDF)		juice (CBW)		juice (SM)	
juice	1,00000	juice	1,00000	juice	1,00000	juice	1,00000
corn	0,80778	corn	0,61136	lemon	0,66858	orange	0,82392
ingredients	0,74564	cleaning	0,58827	combier	0,61004	lemon	0,78669
grateful	0,73979	grateful	0,58400	strawberry	0,60617	strawberry	0,75449
orange	0,73827	rub	0,57677	ginger	0,59856	ginger	0,73977
exercise	0,69748	cook	0,57307	vodka	0,58494	combier	0,72131
lime	0,67115	kitchen	0,57233	orange	0,57769	lime	0,70997
melted	0,66441	sandwich	0,56862	tbsp	0,48444	coctail	0,70628
restaurants	0,64457	orange	0,56707	cocktail	0,47850	vodka	0,70450
sweet	0,63936	ingredients	0,56150	cucumber	0,47739	ingredients	0,68134
bread	0,63936	bread	0,55359	honey	0,45942	gnocco	0,67941

Pozn. Slovo *tbsp* je zkratka pro *table spoon* (česky: polévková lžíce), což je měrná jednotka při vaření.

mnoho slov z téhož dokumentu bude mít velkou podobnost a zastíní tak ostatní slova (viz Tabulka 9). Jelikož v experimentech nebylo nepoužito velké množství dokumentů, nebyla tato matice již dále využívána.

Druhý typ matice je založen na informaci z okolí slova, což pomáhá upřesnit významy slov. Ve výsledcích lze vidět, že byla zahrnuta slova pro vyjádření různého ovoce a také míchaných nápojů, což více odpovídá reálnému použití než v předchozích maticích. Rozhodnout o tom, která z těchto dvou matic odpovídá přesnějšímu významu slova *juice* je velmi subjektivní, ale v tomto konkrétním případě obsahovala matice SM nejlepší výsledky pro hledané slovo.

Při tvorbě CBW slouží dokumenty pouze jako oddělovače textů. Tímto způsobem nedojde k problému jako u matic, které obsahují dokumenty (viz Tabulka 9).

Další část experimentů se zaměřuje na použití operace NOT pro upřesnění dotazu. V případech, kdy se hledá určitý význam slova, ale výsledky ukazují na jiný, je potřeba tyto ostatní významy odstranit. V praxi se ukázalo, že nelze hledané slovo negovat pouze tím slovem, které se jeví jako nejvhodnější, nýbrž tím, které se objevuje ve výsledcích před negací. Příkladem by mohlo být slovo *apple* ve významu ovoce. Aby bylo dosaženo správných výsledků lze výraz negovat slovem *computer*, který zajistí, že ve výsledcích se nebudou objevovat slova spjata s firmou Apple. Nicméně je nutné brát ohled na to, o čem píší články, ze kterých je bráno okolí hledaného slova. Tabulka 10 ukazuje, že výsledky se téměř nezměnily. To je způsobeno tím, že v článcích není tolik zmínek o *apple computer*, ale spíše o *apple pay*, což je technologie firmy Apple, která je spojená s bankovnictvím a platbami. Ve výsledcích se proto objevuje i mnoho pojmů z tohoto oboru. Ve většině případů není možné určit předem jakým slovem negovat, aby byly zajištěny požadované výsledky.

Pro jednoduchou negaci byl zvolen příklad se slovem *lemon* (viz Tabulka 11). Ve výsledcích

Tabulka 9: Výsledky vyhledávání slova *chili* pro T-D, TF-IDF, CBW a SM

chili (T-D)		chili (TF-IDF)		chili (CBW)		chili (SM)	
guzman	1,00000	tortilla	1,00000	chili	1,00000	chili	1,00000
ricotta	1,00000	mohan	1,00000	turmeric	0,91953	cumin	0,93870
pancakes	1,00000	reddy	1,00000	cumin	0,91754	clogging	0,93291
brunch	1,00000	minimalist	1,00000	clogging	0,90350	turmeric	0,93115
sountornsorn	1,00000	bajji	1,00000	uggani	0,90096	uggani	0,91589
lid	1,00000	stuffed	1,00000	vegetable	0,89274	vegetable	0,90863
joshi	1,00000	cosmopolitan	1,00000	shredded	0,88198	shredded	0,89659
prachi	1,00000	fatty	1,00000	paprika	0,87218	paprika	0,88421
tejada	1,00000	wrap	1,00000	coriander	0,85509	onion	0,86844
wrap	1,00000	tejada	1,00000	presto	0,83896	fridge	0,86338
cosmopolitan	1,00000	prachi	1,00000	moham	0,83358	fried	0,83020

Pozn. Slova *turmeric, cumin, coriander* jsou druhy koření. Slovo *uggani* je název kořeněného jídla.

Tabulka 10: Výsledky vyhledávání slova *apple* negované slovem *computer* pro CBW a SM

CBW				SM			
apple		apple NOT computer		apple		apple NOT computer	
apple	1,00000	apple	0,99938	apple	1,00000	apple	0,94085
pay	0,94702	pay	0,94615	transaction	0,97431	chinese	0,91827
transaction	0,93411	transaction	0,93354	pay	0,97211	transaction	0,90746
bank	0,90657	bank	0,90611	chinese	0,96195	anz	0,90536
anz	0,87854	anz	0,87837	bank	0,96130	pay	0,90505
adoption	0,85299	derive	0,85208	anz	0,94322	baulked	0,90148
derive	0,85232	adoption	0,85188	adoption	0,94038	bank	0,90072
fanfare	0,84638	fanfare	0,84620	china	0,94018	fanfare	0,88397
china	0,82398	china	0,82306	fanfare	0,93073	lgpay	0,87561
payment	0,77433	payment	0,77394	debit	0,92516	payment	0,87534

Tabulka 11: Výsledky vyhledávání slova *lemon* pro CBW a SM

Content Bearing Words				Matice podobnosti			
lemon		lemon NOT zest		lemon		lemon NOT zest	
lemon	1,00000	combier	0,71638	lemon	1,00000	combier	0,73170
zest	0,81416	vodka	0,69831	zest	0,90310	vodka	0,71985
tbsp	0,75527	strawberry	0,69703	tbsp	0,88969	ginger	0,70778
broth	0,70682	ginger	0,66129	ahead	0,85133	strawberry	0,69725
ingredients	0,69451	lemon	0,58063	ingredients	0,84462	gnocco	0,60464
ahead	0,67384	absorbed	0,53138	strawberry	0,83994	cucumber	0,55202
juice	0,66858	broth	0,52170	tsp	0,82294	honey	0,54002
tsp	0,66686	cucumber	0,51502	juice	0,78669	policemen	0,52099
stir	0,63102	honey	0,51176	stir	0,77330	spring	0,50273
absorbed	0,59770	juice	0,51082	absorbed	0,76580	absorbed	0,47175
butter	0,56904	gnocco	0,48097	blender	0,76060	cocktail	0,46808
creamy	0,54830	cocktail	0,45556	smooth	0,75380	juice	0,44957
strawberry	0,53770	stir	0,44383	butter	0,75226	simmer	0,43116
combier	0,53500	spring	0,42346	creamy	0,71691	lemon	0,42942

Pozn. Slovo *zest* je citronová kůra.

je možné pozorovat, že slova se týkají citronové kůry, tedy přísady, která se většinou používá při pečení. Negace byla provedena slovem *zest*. Výsledky obsahují různá slova spojená s míchanými nápoji jako např. *combier*, *vodka*, *juice* atd. I když slovo *lemon* bylo tímto procesem degradováno, stále je velmi relevantní a jedná se o velmi užitečný nástroj při vyhledávání. Další správný příklad negace je uveden v tabulkách 4 a 7.

Pro lepší výsledky je nutné použít negaci s disjunkcí, která určí více nerelevantních slov. V následujícím příkladu bylo opět zvoleno slovo *jobs*, ve smyslu zakladatele firmy Apple (viz Tabulka 12 a 13). Slovo bylo spojeno s jednotným číslem *job*, což mělo za následek, že výsledky jsou z okruhu práce nebo z článků, která se zabývají školní přípravou žáků na budoucí zaměstnání. Negací slov *teacher* a *employment* bylo dosaženo relevantnějších výsledků, jmenovitě slov *inventor*, *visionary* nebo *Steve*.

Při srovnání obou matic je zřejmé, že obě obsahují téměř totožné výsledky. Matice CBW méně znehodnotila originální slovo, a tudíž by její výsledky mohly být hodnoceny jako lepší. Nicméně, výsledky obou matic ze všech experimentů byly natolik podobné, že nelze jasně určit, která matice lépe popisuje významy slov.

Poslední příklad popisuje nejkomplexnější formu dotazu za použití několika slov negovaných více slovy (viz Tabulka 14). Z předchozích příkladů je zřejmé, že slovo *apple* úzce souvisí se službou Apple Pay. Pokud bychom ovšem chtěli najít slova související s *apple* v jiném významu, například s jablečným koláčem, je nutné původní význam odstranit. Toho lze dosáhnout pomocí negování slov, které mají nejlepší výsledky vyhledávání a úzkou vazbu na Apple Pay. Negací

Tabulka 12: Výsledky vyhledávání slova *jobs* pro CBW

jobs		jobs NOT (teacher OR employment)	
job	1,000000	job	0,521536
clarifying	0,652692	tweaker	0,426300
teacher	0,650218	inventor	0,424626
employment	0,649250	gladwell	0,418633
expensive	0,623093	visionary	0,417579
finley	0,602493	claimants	0,411766
educational	0,588693	outright	0,405106
proposed	0,586628	repub	0,399333
public	0,585301	malcolm	0,398080
schools	0,579483	broken	0,385020
kids	0,574477	windsor	0,384383
accepted	0,573075	tweak	0,373710
student	0,572798	compares	0,364484
position	0,571277	steve	0,364128

Tabulka 13: Výsledky vyhledávání slova *jobs* pro SM

jobs		jobs NOT (teacher OR employment)	
job	1,000000	inventor	0,451353
expensive	0,903839	outright	0,445641
accepted	0,902184	tweaker	0,433352
teacher	0,897613	pupil	0,429110
environment	0,891628	repub	0,420025
position	0,890869	gladwell	0,388859
kids	0,889355	malcolm	0,387938
public	0,881978	broken	0,360743
give	0,879857	visionary	0,336343
challenge	0,878789	bottom	0,320184
earlier	0,877727	job	0,302017
system	0,877349	outperformed	0,301658
put	0,876261	linear	0,300655
highly	0,875266	steve	0,299310

Tabulka 14: Výsledky vyhledávání slov *apple* nebo *pie* pro CBW a SM

(apple OR pie) NOT (pay OR metadata)			
CBW		SM	
pie	0,985182	pie	0,811083
salads	0,543403	salads	0,653256
savoury	0,532406	bakery	0,650614
bakery	0,521158	savoury	0,650460
puddings	0,506198	oven	0,635098
oven	0,495947	brimming	0,615346
muffins	0,493468	rotten	0,607827
yogurts	0,480731	puddings	0,581385
patriotic	0,476785	muffins	0,571573
maximum	0,464285	eadrey	0,570170
rotten	0,458982	bread	0,569096
watermelon	0,457619	maximum	0,566723
jam	0,451596	yogurts	0,563168

Pozn. Slovo *eadrey* je název restaurace.

slovem *pay* ovšem nebylo dosaženo požadovaného výsledku a bylo nutné negaci podpořit dalším vhodným výrazem, v tomto případě slovem *metadata*. Jelikož dvojitá negace nepřinesla kýžený výsledek, byl hledaný výraz podpořen slovem *pie*. Tato operace vedla k tomu, že i když slovo *apple* bylo znehodnoceno, stále bylo dosaženo relevantních výsledků k slovnímu spojení jablečný koláč. Jelikož slovo *pie* je nerelevantní ke negovaným slovům *pay* a *metadata*, jeho výsledky převyšují výsledky od samotného slova *apple*. I přes tuto skutečnost lze stále v tabulce vidět slova jako *rotten* nebo *jam*, která jsou používaná ve spojení s jablkem.

6.4.2 Podobnost s dokumenty metodou pravděpodobnosti

Jedna z částí výzkumu se zabývá podobností dotazu s celým dokumentem. Pro tento typ vyhledávání byly navrženy 2 metody.

První z nich je určená pouze pro matice s dokumenty (T-D a TF-IDF). Vektory slov v těchto maticích popisují pravděpodobnost s jakou se vyskytují v jednotlivých dokumentech. Po použití vyvinutých operací z kvantové logiky by se poté měla změnit i jejich pravděpodobnost.

Prvním experimentem bylo vyhledávání výrazu *apple* v matici T-D (viz Tabulka 15). Z výsledků vyplývá, že hlavní podíl na definici významu tohoto slova má článek pojednávající o technologii Apple Pay. Operace zahrnovala negaci slovy, která se nejvíce objevují ve spojení s touto firmou. Z výsledků by se mohlo zdát, že tato operace nemá větší vliv na dotaz, avšak si lze povšimnout, že i výsledná podobnost je velice nízká. To znamená, že vektor dotazu je odkloněn od nežádoucích slov, ale ty jsou mu stále nejvíce podobná. Z toho lze vyvodit, že slovo *apple* je příliš závislé na těchto článcích a musela by být provedena negace více slovy. Druhou možností

Tabulka 15: Výsledky vyhledávání dokumentů podle slova *apple* metodou pravděpodobnosti z matice T-D

apple			apple NOT (pay OR iphone OR ipad)		
ID	Popis	Sim	ID	Popis	Sim
8591242	Apple Pay	0,800	8111244	iPhone apps	0,383
4861241	Tech disapointments	0,320	8591242	Apple Pay	0,342
8111244	iPhone apps	0,288	4861241	Tech disapointments	0,336
791240	New iPad	0,288	411242	Apple new smarthone	0,154
411242	Apple new smarthone	0,160	941242	Reagan speech	0,139
941242	Reagan speech	0,128	2851241	Phone bursts into flames	0,119

Tabulka 16: Výsledky vyhledávání dokumentů podle slova *apple* metodou pravděpodobnosti z matice TF-IDF

apple			apple NOT (pay OR iphone OR ipad)		
ID	Popis	Sim	ID	Popis	Sim
8111244	iPhone Apps	0,571	8111244	iPhone Apps	0,620
8591242	Apple Pay	0,487	411242	Apple new smartphones	0,301
791240	New iPad	0,413	2851241	Phone bursts into flames	0,228
411242	Apple new smartphone	0,409	4861241	Tech disapointments	0,136
4861241	Tech disapointments	0,159	14251240	Dinner for honor 9/11	0,129
2851241	Phone bursts into flames	0,133	8031242	Pope Francis car	0,119

je doplnit testovací data o více článků, kde se hledané slovo objevuje v jiném kontextu, neboť nyní tento vektor obsahuje příliš málo informací.

Druhý experiment byl totožný s prvním, ovšem vyhledávání bylo provedeno v maticí TF-IDF (viz Tabulka 16). I zde jsou výsledky velmi podobné. Při operaci negace bylo dosaženo lepších výsledků, i když hodnoty podobnosti jsou velmi nízké. Jako příklad by mohl být zmíněn článek *Dinner for honor 9/11*, který pojednává o večeři v americké restauraci na počest příslušníků zásahových jednotek. Tento článek zmiňuje výraz jablečný koláč a zároveň se v něm vyskytuje výraz *Big Apple* jako označení pro město New York.

Jelikož funkce vyhledávání v obou případech ohodnotila články, kde se hledané slovo nevyskytuje hodnotou 0, dalo by se usuzovat, že v tomto typu vyhledávání přicházíme o abstraktní složku významu slova. Pokud je totiž zmíněn mobilní telefon od společnosti Apple v jednom článku a v druhém popisujeme mobilní telefon společnosti Samsung, dalo by se očekávat, že podobnost mezi nimi nulová nebude. Z tohoto důvodu tyto matice lze považovat za nepřesné při hledání významu slov.

Tabulka 17: Výsledky vyhledávání dokumentů podle slova *apple* metodou množiny slov z CBW

apple			apple NOT (pay OR metadata)		
ID	Popis	Sim	ID	Popis	Sim
2491242	Microsoft ASP.NET	0,555	3671241	Recipes	0,067
3671241	Recipes	0,402	991244	Nova Scotia workers	0,066
51243	Drones in Hollwywood	0,314	51243	Drones in Hollywood	0,055
171240	T-Mobile USA	0,305	4861241	Tech disapointments	0,030
4811243	Apple iPhone 6	0,210	791240	New iPad	0,024
791240	New iPad	0,187	4481243	Digital security risks	0,015

6.4.3 Podobnost s dokumenty metodou množiny slov

Druhá navržená metoda vyhledávání dotazu v dokumentech závisí na využití vzorce podobnosti vektoru s množinou vektorů popsaný v kapitole 3.4.2. V těchto příkladech byly použity matice CBW a SM. Při vyhledávání je porovnáván dotaz s podprostorem vektorů slov, který odpovídá slovníku jedinečných slov dokumentu. Takto je dotaz srovnán se všemi dokumenty.

V tabulce 17 jsou výsledky vyhledávání z matice CBW. Pro slovo *apple* se již neobjevují pouze výsledky pro počítačovou firmu, ale také článek s recepty, kde se objevuje jablko jako ingredience. Dokonce se objevují i výsledky, které slovo *apple* neobsahují, ale týkají se internetových technologií (*Microsoft ASP.NET*) a mobilních operátorů (*T-Mobile USA*), a tak z části můžeme hovořit o relevantních výsledcích. Všechny ostatní články v tabulce již hledané slovo obsahují.

Negace slova zde funguje stejně jako ve vyhledávání jednotlivých termínů, ovšem nelze ho negovat na základě toho, jaká slova dokument obsahuje, ale podle významu samotného slova. Dalo by se předpokládat, že pokud se neobjeví článek o Apple Pay v prvních pozicích vyhledávání, znamená to, že slovo není tolik podobné s termínem *pay*. Ukázalo se, že porovnáváním cele množiny z dokumentu, lze dostat jiné výsledky, ale význam původního slova zůstává stejný jako v porovnávání jednotlivých slov (viz Tabulka 12). Nejlepších výsledků bylo dosaženo použitím stejné negace jako v předchozích experimentech (výraz $appleNOT(payORmetadata)$). Proto pokud by se podle intuice negovalo například slovem *Microsoft*, výsledný vektor by se nezměnil natolik, aby se změnil význam původního slova *apple*.

Výsledky v tabulce 17 ukazují, že tato negace funguje relativně efektivně, jelikož články, které pojednávají o technologiích se posunuly na nižší pozice, a naopak se zlepšilo hodnocení ostatních relevantních článků. Opět si lze všimnout, že tato operace má za následek snížení míry podobnosti. Z výsledků, které jsou nejvíce relevantní, by mohl být zmíněn článek *Nova Scotia workers*, který pojednává o zaměstnancích v jablečných sadech.

Pro srovnání byla použita matice SM (viz Tabulka 18), kde jsou výsledky téměř totožné, ale hodnoty podobnosti jsou daleko vyšší než u matice CBW. V případě negování se tyto hodnoty projeví naopak daleko nižší, ovšem výsledky jsou stále relevantní. Pokud by byly tabulky

Tabulka 18: Výsledky vyhledávání dokumentů podle slova *apple* metodou množiny slov z SM

apple			apple NOT (pay OR metadata)		
ID	Popis	Sim	ID	Popis	Sim
2491242	Microsoft ASP.NET	0,837	3671241	Recipes	0,031
3671241	Recipes	0,715	8571244	The Roger award	0,029
171240	T-Mobile USA	0,658	51243	Drones in Hollywood	0,025
4811243	Apple iPhone 6	0,589	3741243	Christmas donation	0,008
791240	New iPad	0,588	941242	Regan speech	0,007
4831242	Plastic waste	0,406	991244	Nova Scotia workers	0,006

výsledků normalizovány, byly by hodnoty ve všech případech velmi podobné. V tomto případě se negování projevilo také lépe, neboť ve výsledcích je více článků, které souvisejí s jablkem namísto technologií firmy Apple. Jmenovitě jde o články *Christmas donation*, který zmiňuje vánoční tradice krájení jablek a pití moštu nebo například *Reagan speech*, který pojednává o proslovu bývalého prezidenta Spojených států amerických Ronalda Reagana, který slovo zmiňuje ve větě *"We are outperformed when comparing apples to apples."*. Tím dává tomuto slovu zcela nový význam, jelikož se nemyslí přímý význam jako ovoce, ale jde pouze o slovní obrat.

Celkově bych tuto metodu vyhledávání hodnotil pozitivně, avšak má své velké nevýhody, které by dle mého názoru bylo možné odstranit větší databází slov i článků.

7 Závěr

Tato diplomová práce měla za úkol zjistit jaké využití má kvantová informace při zpracování textu. Kvantová logika zde byla interpretována přes operace negace a disjunkce. Formou několika experimentů bylo vyzkoušeno několik metod při zpracování textu a následně vliv operací (negace, disjunkce) při vyhledávání dotazu. Hlavním účelem vyhledávání bylo najít význam dotazovaného slova.

První experiment, při kterém bylo zjišťováno, jaká je účinnost filtrování při procesu automatické extrakce slov, byl proveden pro dva typy filtrů (stoplist a Porter Stemmer). Ukázalo se, že použití SL má velmi pozitivní vliv na výsledky vyhledávání, neboť jsou odstraněna neplnovýznamová slova. Ve výsledcích vyhledávání jsou nalezeny pouze plnovýznamová slova, která umožňují vyjádřit význam dotazu. Proto je tato metoda téměř nutností pro vyhledávání významů slov. Stematizační algoritmus PS přetváří slovo do jeho základního tvaru, a tudíž byl zaznamenán velký úbytek unikátních slov, která by jinak byla nadbytečná. Ovšem použití tohoto filtru je spekulativní, neboť může mít za následek spojení slov, která mají odlišný význam před jejich převodem do základního tvaru, a poté je obtížné tyto významy opět rozlišit. Na druhou stranu tento filtr redukuje velké množství unikátních slov (přibližně 30%), což má velký vliv na reálný čas výpočtů.

Další experimenty se zabývaly samotným vyhledáváním dotazů. Pro vyhodnocení výsledků neexistuje žádná míra, která by přesně určovala jejich správnost, proto mohou být výsledky diskutabilní a záleží jak jsou prezentovány.

Nejprve bylo vyzkoušeno vyhledávání na základě podobnosti dotazu se slovem, kde byly vyzkoušeny operace pro úpravu dotazu a také různé typy matic, ze kterých byly dotazy vytvořeny. Matice typu slovo-dokument (T-D a TF-IDF) se ukázaly jako méně efektivní oproti maticím typu slovo-slovo (CBW a SM) při hledání významu slova, neboť jsou omezeny počtem dokumentů a informacemi v nich. Toto tvrzení by ovšem bylo potřeba ověřit v mnohem větším počtu dokumentů. Matice typu slovo-slovo nejsou omezeny samotnými dokumenty, ale je bráno v potaz okolí slova. Výsledky ukazují, že lépe popisují význam slova a proto se zde také lépe projevují účinky operace negace, která většinou efektivně obracela význam slova požadovaným směrem. Při srovnání matic CBW a SM nelze s jistotou říct, která lépe popisuje význam slova, jelikož jejich výsledky byly velmi podobné. Je potřeba přihlédnout na jejich časovou složitost při jejich vytváření, z které vyplývá, že pro praktické použití je dostačující matice CBW, neboť matice SM má velmi velkou časovou složitost při jejím výpočtu, ale je také potřeba brát v potaz dobu pro výpočet matice, ze které SM vychází.

Druhý typ vyhledávání je zaměřen na podobnost dotazu s celým dokumentem, pro které byly navrhnuty dvě metody. První metoda využívá vektorové reprezentace dokumentů v maticích T-D a TF-IDF. Jednotlivé složky z vektoru slov z těchto matic znázorňují pravděpodobnost pro každý dokument. Při testování byly vyzkoušeny různé typy dotazů a výsledky vyhledávání odpovídaly dokumentům, které hledané slovo obsahovaly. Avšak operace negace nebyla tolik

efektivní, jak by se dalo očekávat. To by ovšem mohlo být způsobeno menším počtem dokumentů a tudíž vektor dotazu neobsahuje tolik informací, aby bylo možné takto dokumenty vyhledávat. Velkou nevýhodou těchto matic bylo, že pokud slovo nebylo obsaženo v dokumentu, nastala zde nulová podobnost, což je nežádoucí při hledání významu slova. Matice T-D i TF-IDF podávaly podobné výsledky. Druhá metoda vyhledávání dokumentů byla zaměřena na podobnost dotazu s množinou slov. Zde byly dokumenty reprezentovány jako matice, která byla vytvořena ze seznamu unikátních slov, které dokument obsahuje. Poté byla vypočítána podobnost dotazu s touto maticí. Pro tuto metodu bylo možné využít matice typu slovo-slovo. Ukázalo se, že tato metoda lépe uchovává význam slova než metoda předchozí. Výsledky sice nebyly naprosto ideální, ale byly zde vyhodnoceny i dokumenty, které hledané slovo neobsahovaly, avšak jejich celkové téma úzce souviselo s dotazem. V případě použití negace v dotazu dopadly výsledky pro matici SM o něco lépe než v případě CBW.

Experimenty byly provedeny na relativně malém počtu dokumentů, proto by bylo výhodné tyto pokusy vyzkoušet i na větším množství, kde by dle mého názoru došlo k lepším výsledkům vyhledávání. Tyto metody by mohly pomoci při vývoji samoučících systémů, které se zabývají zpracováním přirozeného jazyka, jelikož slovník významu slov je získáván samotným zpracováním textu a proto ho není nutné zadávat. Nabízí se zde i prostor pro další typy experimentů, například hledání skupin slov, které odpovídají jednomu významu. Zde bylo možné použít matici SM při hierarchickém shlukování.

Literatura

[1] M. Hayashi, *Quantum Information: An Introduction*, Springer, 2006.

[2] SCHWARZ, Josef. *Současný stav a trendy automatické indexace dokumentů: přehledová studie.* Ikaros [online], 2003, 7(3). ISSN 1212-5075. Dostupné z: `http://www.ikaros.cz/node/1300`.

[3] D. Widdows and S. Peters. *Word Vectors and Quantum Logic: Experiments with negation and disjunction.* Eighth Mathematics of Language Conference, Bloomington, Indiana, June 20-22, 2003, pages 141-154.

[4] MANNING, Christopher D., Prabhakar RAGHAVAN a Hinrich SCHÜTZE. *Introduction to information retrieval.* Cambridge: Cambridge University Press, 2008. ISBN 978-0-521-86571-5.

[5] ANSEN, B. J.; RIEH, S. The Seventeen Theoretical Constructs of Information Searching and Information Retrieval. *Journal of the American Society for Information Sciences and Technology*. 2010, roč. 61, čís. 8, s. 1517-1534. [online]. Dostupné z: `https://faculty.ist.psu.edu/jjansen/academic/jansen_theoretical_constructs.pdf`

[6] Martin Porter *Porter Stemmer Algorithm* [online]. Dostupné z: `https://tartarus.org/martin/PorterStemmer/`

[7] Stránky Mathnet.Numerics *Math.NET Numerics* [online]. Dostupné z: `https://numerics.mathdotnet.com/`

[8] Stránky corpusdata.org *Full-text data from the BYU corpora (COCA, COHA, GloWbE, NOW, Wikipedia, Spanish* [online]. Dostupné z: `https://www.corpusdata.org/`

A Příloha na CD

- **/APLIKACE** - Zdrojový kód a dokumentace systému
- **/DATA** - Zdrojová data
  - Dataset
  - Seznam stopslov
- **/DOKUMENT** - Elektronická podoba diplomové práce